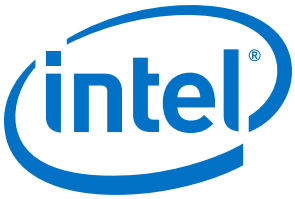


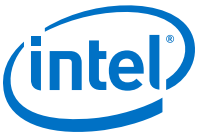
Intel® Omni-Path IP and LNet Router

Design Guide

Rev. 10.0

January 2020

Doc. No.: H99668, Rev.: 10.0



Revision History

Date	Revision	Description
January 2020	10.0	Updates for this revision include: • Updated links to Intel® Omni-Path documentation set in Intel® Omni-Path Documentation Library. • Updated Single Router on page 17. • Updated software download link in RHEL* and SLES* Installation.
October 2019	9.0	Updates for this revision include: • Updated Preface to include new Best Practices. • Added new section, Prerequisites, to Router Redundancy/Failover with VRRP v3. • Added new section, IP Routing Guidance.
December 2018	8.0	Updates for this revision include: • Updated Introduction • Updated Linux IP Router, renamed and reorganized several subsections • Updated Hardware Requirements • Updated Software Requirements, renamed and reorganized several subsections • Updated Installation Steps • Reordered MTU material and added new subsections: Concerning Maximum Transmission Units (MTUs) • Reordered and updated the OPA Interface Tuning section and added new material for Accelerated Datagram Mode and Datagram Mode or Connected Mode - Which is Better? • Updated Ethernet Interface Tuning • Updated Measuring Baseline IP Performance • Updated LNet Router Deployment Checklist • Updated Overview of Configuring LNet • Updated Software Compatibility • Updated Example 1: Legacy Storage with InfiniBand* Card Connected to New Compute Nodes Using Intel OPA • Updated Configure LNet Routers (Example 1) • Updated Example 2: New Storage with Intel OPA Connected to Legacy Compute Nodes on InfiniBand* Cards • Updated Configure LNet Routers (Example 2)
September 2018	7.0	Updates for this revision include: • Rewrote and reorganized the bulk of Performance Benchmarking and Tuning. • Added a note to LNet Troubleshooting to specify Lustre* version. • Added the Preface. • Updated the Introduction. • Updated Virtual Routers with Fault-Tolerance. • Updated Multiple Active Routers with Fault Tolerance.
April 2018	6.0	• Updated the example figures. • Updated the LNet material in Section 3. • Removed all references to Load Balancing using IPVS. • Changed build keepalived to pre-built keepalived.

continued...

Date	Revision	Description
		• Appendix: grat arp workaround for keepalived.
May 2017	5.0	Changed title of document to include "IP".
January 2017	4.0	Updated Section 3.
August 2016	3.0	Minor correction in Section 2.5.2.1 Adding Static Routes.
February 2016	2.0	• Added LNet Router content. • Added RHEL 7.2 SLES 12 SP1.
November 2015	1.0	Initial release.

Contents

Figures

Tables

Preface

This manual is part of the documentation set for the Intel® Omni-Path Fabric (Intel® OP Fabric), which is an end-to-end solution consisting of Intel® Omni-Path Host Fabric Interfaces (HFIs), Intel® Omni-Path switches, and fabric management and development tools.

The Intel® OP Fabric delivers the next generation, High-Performance Computing (HPC) network solution that is designed to cost-effectively meet the growth, density, and reliability requirements of large-scale HPC clusters.

Both the Intel® OP Fabric and standard InfiniBand* (IB) are able to send Internet Protocol (IP) traffic over the fabric, or *IPoFabric*. In this document, however, it may also be referred to as *IP over IB* or *IPoIB*. From a software point of view, IPoFabric behaves the same way as IPoIB, and in fact uses an ib_ipoib driver to send IP traffic over the ib0/ib1 ports.

Intended Audience

The intended audience for the Intel® Omni-Path (Intel® OP) document set is network administrators and other qualified personnel.

Intel® Omni-Path Documentation Library

Intel® Omni-Path publications are available at the following URL, under *Latest Release Library*:

https://www.intel.com/content/www/us/en/design/products-and-solutions/networking-and-io/fabric-products/omni-path/downloads.html

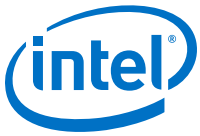
Use the tasks listed in this table to find the corresponding Intel® Omni-Path document.

Task	Document Title	Description
Using the Intel® OPA documentation set	*Intel® Omni-Path Fabric Quick Start Guide*	A roadmap to Intel's comprehensive library of publications describing all aspects of the product family. This document outlines the most basic steps for getting your Intel® Omni-Path Architecture (Intel® OPA) cluster installed and operational.
Setting up an Intel® OPA cluster	*Intel® Omni-Path Fabric Setup Guide*	Provides a high level overview of the steps required to stage a customer-based installation of the Intel® Omni-Path Fabric. Procedures and key reference documents, such as Intel® Omni-Path user guides and installation guides, are provided to clarify the process. Additional commands and best known methods are defined to facilitate the installation process and troubleshooting.

continued...

Task	Document Title	Description
Installing hardware	*Intel® Omni-Path Fabric Switches Hardware Installation Guide*	Describes the hardware installation and initial configuration tasks for the Intel® Omni-Path Switches 100 Series. This includes: Intel® Omni-Path Edge Switches 100 Series, 24 and 48-port configurable Edge switches, and Intel® Omni-Path Director Class Switches 100 Series.
	Intel® Omni-Path Host Fabric Interface Installation Guide	Contains instructions for installing the HFI in an Intel® OPA cluster.
Installing host software Installing HFI firmware Installing switch firmware (externally-managed switches)	*Intel® Omni-Path Fabric Software Installation Guide*	Describes using a Text-based User Interface (TUI) to guide you through the installation process. You have the option of using command line interface (CLI) commands to perform the installation or install using the Linux* distribution software.
Managing a switch using Chassis Viewer GUI Installing switch firmware (managed switches)	*Intel® Omni-Path Fabric Switches GUI User Guide*	Describes the graphical user interface (GUI) of the Intel® Omni-Path Fabric Chassis Viewer GUI. This document provides task-oriented procedures for configuring and managing the Intel® Omni-Path Switch family. Help: GUI embedded help files
Managing a switch using the CLI Installing switch firmware (managed switches)	*Intel® Omni-Path Fabric Switches Command Line Interface Reference Guide*	Describes the command line interface (CLI) task information for the Intel® Omni-Path Switch family. Help: -help for each CLI
Managing a fabric using FastFabric	*Intel® Omni-Path Fabric Suite FastFabric User Guide*	Provides instructions for using the set of fabric management tools designed to simplify and optimize common fabric management tasks. The management tools consist of Text-based User Interface (TUI) menus and command line interface (CLI) commands. Help: -help and man pages for each CLI. Also, all host CLI commands can be accessed as console help in the Fabric Manager GUI.
Managing a fabric using Fabric Manager	*Intel® Omni-Path Fabric Suite Fabric Manager User Guide*	The Fabric Manager uses a well defined management protocol to communicate with management agents in every Intel® Omni-Path Host Fabric Interface (HFI) and switch. Through these interfaces the Fabric Manager is able to discover, configure, and monitor the fabric.
	Intel® Omni-Path Fabric Suite Fabric Manager GUI User Guide	Provides an intuitive, scalable dashboard and set of analysis tools for graphically monitoring fabric status and configuration. This document is a user-friendly alternative to traditional command-line tools for day-to-day monitoring of fabric health. Help: Fabric Manager GUI embedded help files
Configuring and administering Intel® HFI and IPoIB driver Running MPI applications on Intel® OPA	*Intel® Omni-Path Fabric Host Software User Guide*	Describes how to set up and administer the Host Fabric Interface (HFI) after the software has been installed. The audience for this document includes cluster administrators and Message-Passing Interface (MPI) application programmers.
Writing and running middleware that uses Intel® OPA	*Intel® Performance Scaled Messaging 2 (PSM2) Programmer's Guide*	Provides a reference for programmers working with the Intel® PSM2 Application Programming Interface (API). The Performance Scaled Messaging 2 API (PSM2 API) is a low-level user-level communications interface.

continued...



Task	Document Title	Description
Optimizing system performance	*Intel® Omni-Path Fabric Performance Tuning User Guide*	Describes BIOS settings and parameters that have been shown to ensure best performance, or make performance more consistent, on Intel® Omni-Path Architecture. If you are interested in benchmarking the performance of your system, these tips may help you obtain better performance.
Designing an IP or LNet router on Intel® OPA	*Intel® Omni-Path IP and LNet Router Design Guide*	Describes how to install, configure, and administer an IPoIB router solution (Linux* IP or LNet) for inter-operating between Intel® Omni-Path and a legacy InfiniBand* fabric.
Building Containers for Intel® OPA fabrics	*Building Containers for Intel® Omni-Path Fabrics using Docker* and Singularity* Application Note*	Provides basic information for building and running Docker* and Singularity* containers on Linux*-based computer platforms that incorporate Intel® Omni-Path networking technology.
Writing management applications that interface with Intel® OPA	*Intel® Omni-Path Management API Programmer's Guide*	Contains a reference for programmers working with the Intel® Omni-Path Architecture Management (Intel OPAMGT) Application Programming Interface (API). The Intel OPAMGT API is a C-API permitting in-band and out-of-band queries of the FM's Subnet Administrator and Performance Administrator.
Using NVMe* over Fabrics on Intel® OPA	*Configuring Non-Volatile Memory Express* (NVMe*) over Fabrics on Intel® Omni-Path Architecture Application Note*	Describes how to implement a simple Intel® Omni-Path Architecture-based point-to-point configuration with one target and one host server.
Learning about new release features, open issues, and resolved issues for a particular release	*Intel® Omni-Path Fabric Software Release Notes*	
	Intel® Omni-Path Fabric Manager GUI Release Notes	
	Intel® Omni-Path Fabric Switches Release Notes (includes managed and externally-managed switches)	
	Intel® Omni-Path Fabric Unified Extensible Firmware Interface (UEFI) Release Notes	
	Intel® Omni-Path Fabric Thermal Management Microchip (TMM) Release Notes	
	Intel® Omni-Path Fabric Firmware Tools Release Notes	

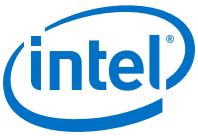
How to Search the Intel® Omni-Path Documentation Set

Many PDF readers, such as Adobe* Reader and Foxit* Reader, allow you to search across multiple PDFs in a folder.

Follow these steps:

1. Download and unzip all the Intel® Omni-Path PDFs into a single folder.
2. Open your PDF reader and use **CTRL-SHIFT-F** to open the Advanced Search window.
3. Select **All PDF documents in...**
4. Select **Browse for Location** in the dropdown menu and navigate to the folder containing the PDFs.
5. Enter the string you are looking for and click **Search**.

Use advanced features to further refine your search criteria. Refer to your PDF reader Help for details.



Cluster Configurator for Intel® Omni-Path Fabric

The Cluster Configurator for Intel® Omni-Path Fabric is available at: http://www.intel.com/content/www/us/en/high-performance-computing-fabrics/omni-path-configurator.html.

This tool generates sample cluster configurations based on key cluster attributes, including a side-by-side comparison of up to four cluster configurations. The tool also generates parts lists and cluster diagrams.

Documentation Conventions

The following conventions are standard for Intel® Omni-Path documentation:

- *Note:* provides additional information.
- ***Caution:*** indicates the presence of a hazard that has the potential of causing damage to data or equipment.
- ***Warning:*** indicates the presence of a hazard that has the potential of causing personal injury.
- Text in blue font indicates a hyperlink (jump) to a figure, table, or section in this guide. Links to websites are also shown in blue. For example:

  See License Agreements on page 13 for more information.

  For more information, visit www.intel.com.
- Text in **bold** font indicates user interface elements such as menu items, buttons, check boxes, key names, key strokes, or column headings. For example:

  Click the **Start** button, point to **Programs**, point to **Accessories**, and then click **Command Prompt**.

  Press **CTRL+P** and then press the **UP ARROW** key.
- Text in `Courier` font indicates a file name, directory path, or command line text. For example:

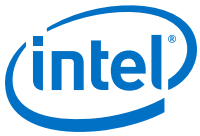
  Enter the following command: `sh ./install.bin`
- Text in *italics* indicates terms, emphasis, variables, or document titles. For example:

  Refer to *Intel® Omni-Path Fabric Software Installation Guide* for details.

  In this document, the term *chassis* refers to a managed switch.

Procedures and information may be marked with one of the following qualifications:

- **(Linux)** – Tasks are only applicable when Linux* is being used.
- **(Host)** – Tasks are only applicable when Intel® Omni-Path Fabric Host Software or Intel® Omni-Path Fabric Suite is being used on the hosts.
- **(Switch)** – Tasks are applicable only when Intel® Omni-Path Switches or Chassis are being used.
- Tasks that are generally applicable to all environments are not marked.



Best Practices

- Intel recommends that users update to the latest versions of Intel® Omni-Path firmware and software to obtain the most recent functional and security updates.
- To improve security, the administrator should log out users and disable multi-user logins prior to performing provisioning and similar tasks.

License Agreements

This software is provided under one or more license agreements. Please refer to the license agreement(s) provided with the software for specific detail. Do not install or use the software until you have carefully read and agree to the terms and conditions of the license agreement(s). By loading or using the software, you agree to the terms of the license agreement(s). If you do not wish to so agree, do not install or use the software.

Technical Support

Technical support for Intel® Omni-Path products is available 24 hours a day, 365 days a year. Please contact Intel Customer Support or visit http://www.intel.com/omnipath/support for additional detail.

1.0 Introduction

An Intel® Omni-Path Architecture (OPA) fabric often requires connectivity to a parallel file system. It is recommended to attach the storage units directly to the OPA fabric using a host fabric interface to enable the use of RDMA for bulk data transfers. When storage cannot be directly attached to the OPA fabric, a gateway/router may be deployed to provide storage connectivity. For Lustre* based filesystems, LNet routers can be used to connect the OPA fabric to another network with the storage units. In other cases, IP routers may be considered.

Figure 1. Direct Fabric Attached Multi-homed Storage

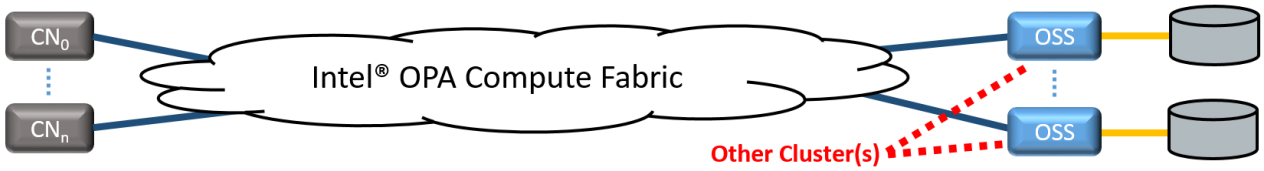


Figure 2. Lustre* (LNet) Routing to InfiniBand* or Ethernet

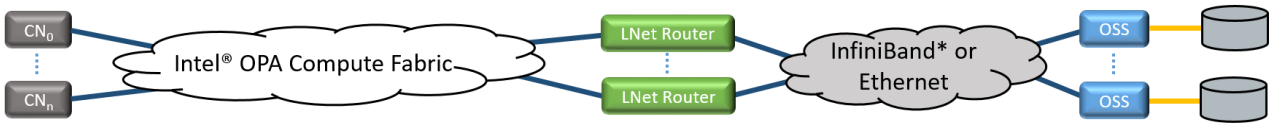
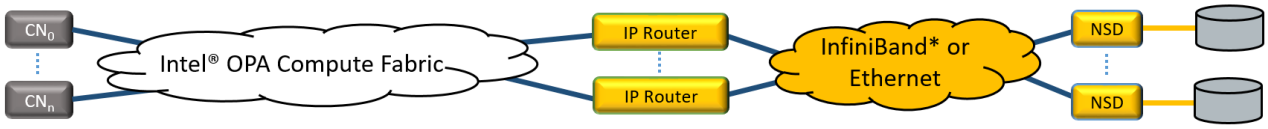
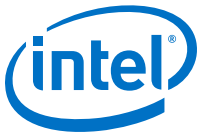


Figure 3. IP Routing to InfiniBand* or Ethernet



This design guide describes several possible routing options:

- **LNet Router** - The LNet router is a node that can forward Lustre* traffic from one type of network to another using the Lustre* stack installed in the node. It is specifically designed to route Lustre* traffic natively from one network type to another in addition to the ability to perform load-balancing and failover. When storage nodes cannot be attached to the OPA fabric with the Lustre* filesystem, an LNet router may be configured to provide necessary connectivity. LNet routing is introduced in LNet Router on page 44.
- **IP Router** - The IP router is a node that can forward IP packets from one type of network to another using a default IP stack in Linux. It can be used to provide IP connectivity between OPA and other network types, when storage nodes cannot be attached to the OPA fabric. Based on card types installed in the IP router node, IP-routing between OPA fabrics and 10/25/40/50/100/200/400 Gbps Ethernet and/or to SDR/DDR/FDR/EDR/HDR InfiniBand* networks using IPoIB may be considered. IP routing, for both OPA-InfiniBand* and OPA-Ethernet, is introduced in Linux IP Router on page 16.

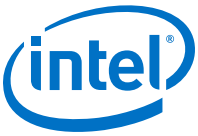


Appendices provide additional information related to firewall and keepalived configuration.

1.1 Special Conventions Used

Special conventions used in this document include:

- # preceding a command indicates the command is to be entered as root.
- $ indicates a command is to be entered as a user.
- <*variable_name*> indicates the placeholder text that appears between the angle brackets is to be replaced with an appropriate value.



2.0 Linux IP Router

This section discusses the use of gateway/router nodes for IP routing between OPA fabrics and existing InfiniBand* (IB) fabrics, over high-speed network connections. The basic design can also be used for IP routing between OPA fabrics and existing Ethernet* networks.

End users include systems administrators, network administrators, cluster administrators and other qualified personnel tasked with coupling the Intel® Omni-Path Fabric to existing IB fabrics or Ethernet networks.

2.1 IP Over Fabric

The traditional method for sending IP traffic over an InfiniBand* fabric is IP over IB or *IPoIB (RFC 4391, 4392)*. In this document, running IPoIB on an Intel® Omni-Path fabric is referred to as IP over Fabric or *IPoFabric*.

IPoFabric supports both datagram mode and connected mode configuration and operation. Starting with the IFS 10.9 Intel® Omni-Path software release, improvements were introduced to make IPoFabric run better in datagram mode. Improvements include:

- A multi-queue network interface for IPoFabric that better leverages existing OPA hardware features to accelerate parallel sends and receives
- The ability to support larger jumbo datagram mode MTU sizes, up to approximately 10K, for improved performance.

These IPoFabric performance improvements can directly result in performance improvement for IP routers, especially when routing to an Ethernet network that employs a Jumbo MTU size.

Though IPoFabric offers features not found in standard IPoIB, a user already used to working with IPoIB should find working with IPoFabric familiar; it is generally configured and operates similarly.

Both IPoIB and IPoFabric provide a standard network device interface to the Linux networking stack. Thus, it is possible to configure a Linux node to act as a gateway/router node for routing IP traffic between an InfiniBand interface and an OPA interface on the same host, named as an IP router in this document. With an IP router, applications can share data and files between IB and OPA fabrics using standard network protocols, like TCP/IP.

In addition to routing TCP/IP packets between an IB and an OPA fabric, the same methodology can be used to route between an OPA fabric and an Ethernet network. The biggest difference is the interface-specific configuration used on the router, but the general configuration and routing setup is similar regardless of the type of the non-OPA network.

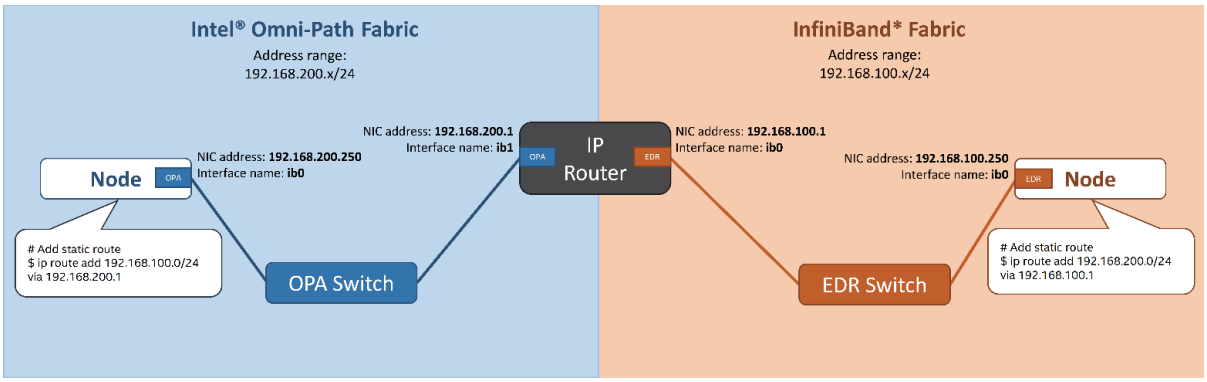
2.2 Illustrative Linux IP Router Use Cases

The following three examples demonstrate some ways in which an IP router, or group of IP routers, may be used to connect an OPA fabric to management or storage on an external network. These cases are illustrative, and are not intended to be exhaustive; other cases may be possible.

2.2.1 Single Router

The figure below illustrates a case with a single router connecting the Intel® Omni-Path fabric to the IB fabric. This is the simplest setup possible to gain connectivity between the Intel® Omni-Path and IB fabrics. There is no fail-over and no load-balancing. This is a typical setup used for testing and development.

Figure 4. Single Router Example

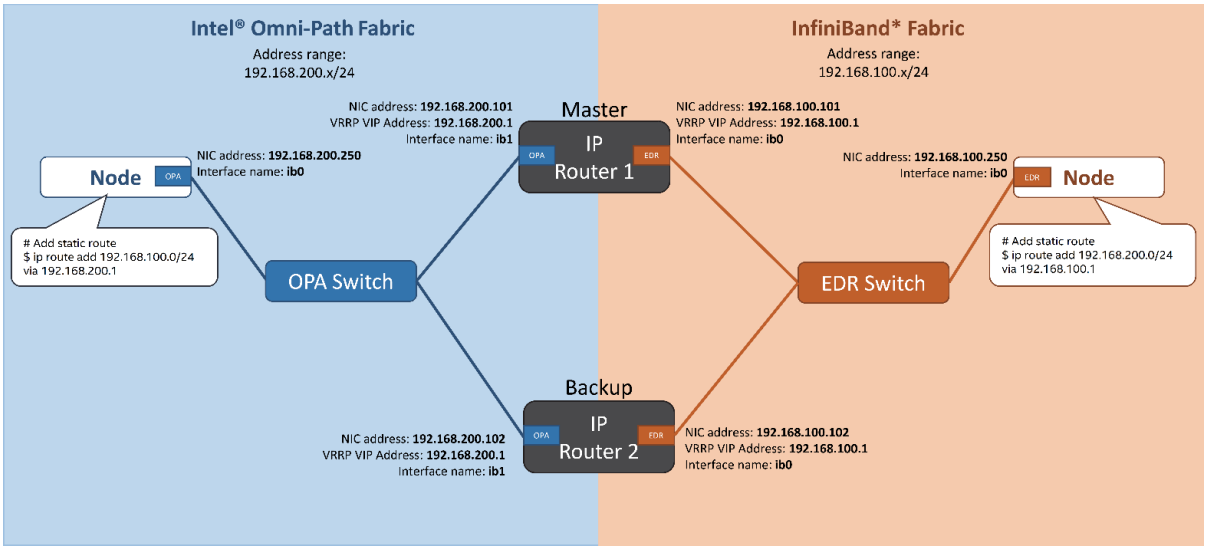


It is possible to increase reliability of a single router configuration by adding a second network interface and using IPoIB bonding in active/standby mode. In the case of an adapter or cable failure, the traffic will use the secondary IPoFabric interface. Refer to IPoIB Bonding section of the *Intel® Omni-Path Fabric Host Software User Guide* for details.

2.2.2 Virtual Routers with Fault-Tolerance

This setup includes two routers running Virtual Router Redundancy Protocol (VRRP) to provide fault-tolerance and fail-over. If a single router fails with this setup, having two routers provides automatic fail-over of the virtual IP address used for routing between the subnets from one physical router node to a different physical router node.

Figure 5. Fault-tolerant Router Pair Example



A variant of this configuration uses two virtual addresses. Each router is master for one of the addresses, with the opposite one being the backup. This allows full utilization of both routers when both are active. A failure of either one moves the failed router's primary virtual address to the remaining router.

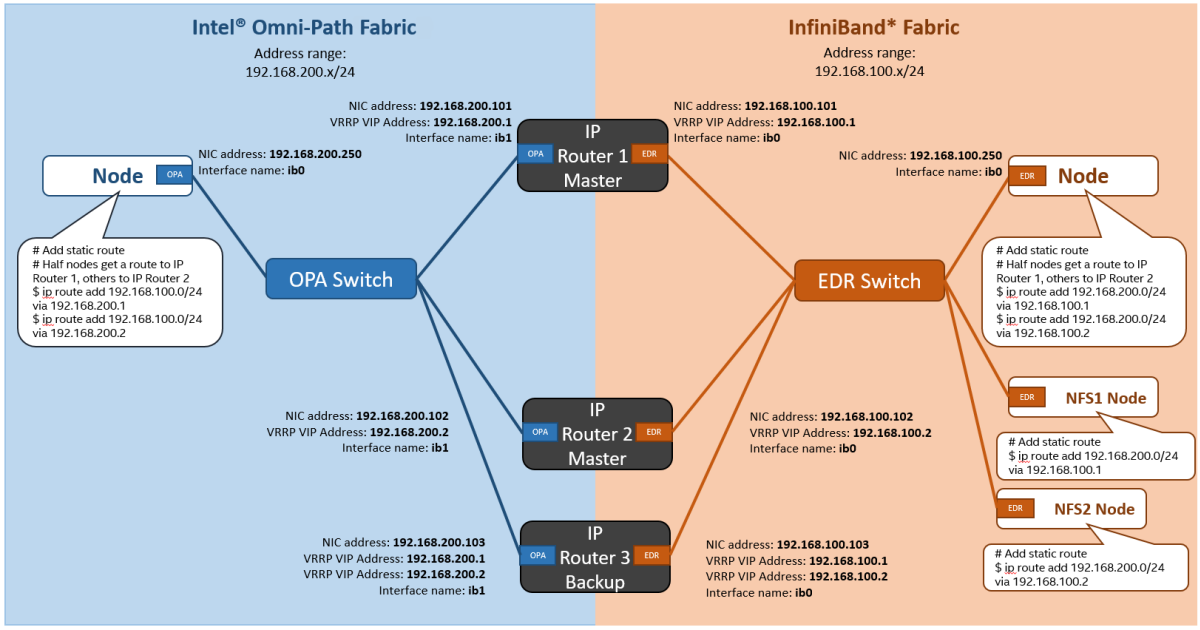
2.2.3 Multiple Active Routers with Fault Tolerance

This setup contains three routers: two masters and one backup. In this scenario the traffic load is distributed across the two master routers to allow for greater aggregate throughput. The routers are on the same IP subnet, and traffic is directed to one router or the other using static routes in the end nodes. As in previous examples, *keepalived* is used to provide redundancy, this time in an N+1 fashion. In this example, router 1 has VIPs 192.168.100.1 and 192.168.200.1, while router 2 has VIPs 192.168.100.2 and 192.168.200.2. The backup router sits in standby mode to accommodate the failure of one or both master routers.

NOTE

There are other schemes that could be used to distribute load across multiple routers; this example is just one scheme.

Figure 6. Load Distribution Across Multiple Routers Example



A variant of this configuration uses three virtual IP addresses. The backup router for the first two virtual addresses can serve as primary router for the third address. One of the first two routers can serve as backup for the third. This allows full utilization of all routers while they are active.

2.3 Hardware Requirements

A server with at least one Intel® Xeon® Processor with eight or more cores, PCIe* Gen3 x16 link interfaces, and a minimum of 8 GB RAM may be required for the minimum testing requirement. For best performance, all IO channels on the memory controller should be populated equally, in a balanced configuration. Systems using dual-port or additional HFI/HCA adapters may be able to take advantage of a second Intel® Xeon® Processor.

The needs of IP routing applications generally differ from the needs of compute applications for performance. For IP routing applications, Intel generally recommends use of servers that are able to run at a higher per-core clock frequency, compared to servers that may have more cores that run at a lower per-core clock frequency.

However, these are simply generalizations, and not specific recommendations. As a knowledgeable reader can appreciate, server and platform technology evolves quickly, and a router's performance will be dependent on many factors. These factors may include types and speed of memory used, types of processor(s), number of sockets, number of cores per socket, clock speeds of cores, link speeds, characteristics of the platform's PCIe and bus architecture, physical interface characteristics, BIOS settings, kernel and driver software versions and tunings, test configuration, packet sizes, and workload characteristics just to name a few. This is not an exhaustive list. Many factors, including contention for available PCIe, interrupt, and memory bandwidth, in addition to availability of suitable card slots in a platform and other factors may limit

the practical benefits of adding additional HFI and/or other network adapter cards to create a router node. For these reasons, it is advised to consult with an applications specialist when planning your router nodes.

When planning for a router, in addition to the platform considerations, additional components need to be considered::

- Intel® Omni-Path Fabric HFI (one or more for each router)
- QDR/FDR/EDR HCA, or 10/25/40/50/100 Gbps Ethernet cards (one, or more, for each router)
- Interface cables to connect to the router's OPA port(s)
- Interface cables to connect to the router's IB/Ethernet port(s)
- Available switch ports to connect those cables to in each routed network
- Rack space and power for the routers

2.4 Software Requirements

2.4.1 BIOS Settings

Setting the system BIOS is an important step in configuring a cluster to provide the best mix of application performance and power efficiency. In the following, we specify settings that should maximize the Intel® Omni-Path Fabric and application performance. Optimally settings similar to these should be used during a cluster bring-up and validation phase to show that the fabric is performing as expected. For the long term, you may want to set the BIOS to provide more power savings, even though that will reduce overall application and fabric performance to some extent.

For BIOS settings, reference the *Intel® Omni-Path Fabric Performance Tuning User Guide.*

2.4.2 RHEL* and SLES* Installation

The Red Hat* Enterprise Linux (RHEL) or SUSE* SPx (SLES) software version used must be on the list of versions supported by Intel® OPA, available from the Intel Resource & Design Center (https://www.intel.com/content/www/us/en/design/products-and-solutions/networking-and-io/fabric-products/omni-path/downloads.html).

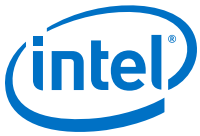
For additional OS software requirements, refer to the *Intel® Omni-Path Fabric Software Release Notes* and *Intel® Omni-Path Fabric Software Installation Guide*.

NOTE

Instructions for CentOS and Scientific Linux are the same as for RHEL.

NOTE

You may be required to disable the firewall and SELinux during installation. Please see Firewall Configuration for VRRP on RHEL on page 63 for additional information.



1. Disable firewalld during the Intel® Omni-Path Basic installation period:

```
# systemctl disable firewalld
# systemctl stop firewalld
```

2. Disable SELinux during the install period:

```
# setenforce 0
```

3. Edit `/etc/sysconfig/selinux` and change `SELINUX=enforcing` to `SELINUX=disabled` to keep SELinux disabled on reboot.

2.4.3 Intel® Omni-Path Software Installation

2.4.3.1 Software Requirements

- Intel® Omni-Path IFS on Intel® Omni-Path node
- Intel® Omni-Path Basic on Coexist routers

When routing between OPA and IB, each fabric requires their own subnet manager, and neither fabric's subnet manager should be run on the router node. The Intel® Omni-Path Fabric Suite Fabric Manager should NOT be enabled on the coexist routers, nor should the IB subnet manager. When installing software, the Intel® Omni-Path Basic install package should be used on the router node because it does not contain the Intel® Omni-Path Fabric Suite Fabric Manager, nor the InfiniBand* opensm service.

NOTE

Unless specifically identified, the instructions for Intel® Omni-Path software installation are for both RHEL* and SLES*.

2.4.3.2 Installation Steps

1. Install the Intel® Omni-Path Basic build on the routers using instructions from the *Intel® Omni-Path Fabric Software Installation Guide* and apply tunings for IPoFabric as recommended by the *Intel® Omni-Path Fabric Performance Tuning User Guide*.

NOTE

To ensure that the mlx4_ib module loads correctly at boot time, add the following line to `/etc/modprobe.d/mlx4.conf`:

```
install mlx4_ib /usr/sbin/modprobe --ignore-install -f mlx4_ib
```

This is necessary to ensure that modprobe skips the check of symbol signatures that are different from IFS-built ib_mad, ib_sa, ib_umad modules sharing the same API.

You should also see the following mlx modules:

```
# lsmod |grep mlx

mlx4_ib               158552  2
mlx4_en               94530  0
vxlan                 37409  1 mlx4_en
ib_sa                  33949  5
rdma_cm,ib_cm,mlx4_ib,rdma_ucm,ib_ipoib
ib_mad                 61179  5 hfi1,ib_cm,ib_sa,mlx4_ib,ib_umad
ptp                    18933  2 igb,mlx4_en
mlx4_core             254286  2 mlx4_en,mlx4_ib
ib_core                88311  12
hfi1,rdma_cm,ib_cm,ib_sa,iw_cm,mlx4_ib,ib_mad,ib_ucm,ib_umad,
ib_uverbs,rdma_ucm,ib_ipoib
```

NOTE

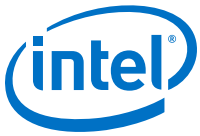
mlx4_ib might be replaced by mlx5_ib on later versions of the software. Consult InfiniBand* vendor documentation for more details.

2. With Intel® Omni-Path software installed and the HFI1 and mlx drivers loaded, confirm that both devices are active on their respective fabrics.

   a. To check the Intel® Omni-Path HFI, run `opainfo`:

```
# opainfo
hfi1_0:1
PortGID:0xfe80000000000000:0011750101574238
   PortState:     Active
   LinkSpeed      Act: 25Gb         En: 25Gb
   LinkWidth      Act: 4            En: 4
   LinkWidthDnGrd ActTx: 4  Rx: 4   En: 3,4
   LCRC           Act: 14-bit       En: 14-bit,16-bit,48-bit
Mgmt: True
   LID: 0x00000006-0x00000006       SM LID: 0x00000001 SL: 0
   QSFP: PassiveCu,   2m TE Connectivity   P/N 2821076-2
Rev B
   Xmit Data:               961 MB Pkts:             13540087
   Recv Data:             28702 MB Pkts:             15932547
   Link Quality: 5 (Excellent)
   Integrity Err:                    0 Err Recovery          0
```

   Notice that the PortState is Active.



b. To check the InfiniBand* HCA, run `ibstat`:

```
# ibstat

CA 'mlx4_0'
    CA type: MT4099
    Number of ports: 1
    Firmware version: 2.34.5000
    Hardware version: 0
    Node GUID: 0x0002c903003d5bf0
    System image GUID: 0x0002c903003d5bf3
    Port 1:
          State: Active
          Physical state: LinkUp
          Rate: 56
          Base lid: 17
          LMC: 0
          SM lid: 5
          Capability mask: 0x02514868
          Port GUID: 0x0002c903003d5bf1
          Link layer: InfiniBand
```

Notice that State is Active.

3. If the Intel® Omni-Path HFI is inactive, confirm that the Intel® Omni-Path Fabric Suite Fabric Manager (opafm) is running on either the Intel® Omni-Path switch or an Intel® Omni-Path node. If the IB HCA is inactive, check that the IB subnet manager (opensm) is running on the IB switch or an IB node.

2.5 Router Configuration

2.5.1 Network Interface Naming Consistency

2.5.1.1 Network Interface Naming Using Explicit Driver Loading

On Linux, the order in which the HFI/HCA drivers are loaded determines which interface name (ib0, ib1 …) is associated with the driver. The network interface configuration files in `/etc/sysconfig/network-scripts` for RHEL* or `/etc/sysconfig/network` for SLES* depend on the appropriate driver being assigned to the correct interface. You can either observe the behavior on your system after the cards have been installed or you can explicitly control the order in which the drivers are installed.

For controlling driver installation, you must first "blacklist" the drivers involved by using a text file located in `/etc/modprobe.d` containing something similar to the following (example is for routing OPA-IB):

```
# Blacklist the InfiniBand/ Intel® Omni-Path drivers to prevent the
# system automatically loading them at startup.
blacklist mlx4_core
blacklist hfi1
```

A script stored in /etc/sysconfig/modules is then used to start the drivers in a particular order:

```
#!/bin/sh
# Influence the order of the interfaces - the first one loaded
# gets ib0 the next ib1, etc.
if [ ! -c /dev/ipath0 ] ; then
    /sbin/modprobe --force mlx4_core > /dev/null 2>&1
fi
if [ ! -c /dev/hfi1 ] ; then
    /sbin/modprobe hfi1 > /dev/null 2>&1
fi
```

2.5.1.2 Network Interface Naming Using udev

Another method for setting device names is through udev. In the following example, you see the /etc/udev/rules.d/70-persistent-ipoib.rules file. The directions in the file are very clear and to the point.

```
>>>>>>
#This is a sample udev rules file that demonstrates how to get udev to
# set the name of IPoIB interfaces to whatever you want. There is a
# 16 character limit on network device names.
#
# Important items to note: ATTR{type}=="32" is IPoIB interfaces, and the
# ATTR{address} match must start with ?* and only reference the last 8
# bytes of the address or else the address might not match on any given
# start of the IPoIB stack
#
# Note: as of rhel7, udev is case sensitive on the address field match
# and all addresses need to be in lower case.
#
ACTION=="add", SUBSYSTEM=="net", DRIVERS=="?*", ATTR{type}=="32",
ATTR{address}=="?*00:11:75:01:01:57:51:59", NAME="ib0"
ACTION=="add", SUBSYSTEM=="net", DRIVERS=="?*", ATTR{type}=="32",
ATTR{address}=="?*00:02:c9:03:00:33:13:81", NAME="ib1"
<<<<<<<
```

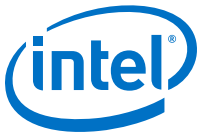
In the previous file example the entry "?*00:11:75:01:01:57:51:59", NAME="ib0" defines the interface itself via the last 8 bytes of the IB/ Intel® Omni-Path GUID, which can be viewed by the output of ip a, and defines the device name mapping you choose for the interface.

Depending on how you have placed the HFI/HCA on the motherboard, you may see that IB1 is listed above IB0 with the output of ip a, but for purposes of scripting the devices will always have the same device name.

Both the explicit driver loading method and the udev method require a working OS and manual intervention, which obviates network installs from using kickstart and other provisioning tools. To guarantee the correct device name assignments you must place the devices in the proper order in the PCI bridges on the motherboard. If this is done for each router before the builds, neither of the previously mentioned modifications are necessary.

2.5.1.3 Configuring Network Devices with nmcli for RHEL*

The Intel® Omni-Path Basic installation will ask if you want to configure the IPoIB network scripts. RHEL* (version 7 or later) no longer uses the ifconfig utility, replacing it with tools from the NetworkManager service. While you can choose to



configure the HFI/IB devices using the Intel® Omni-Path install tool, Intel recommends using the default RHEL* network tools. If the Intel® Omni-Path device is not recognized by the OS (the IP utility is not showing an IB device from the output of `ip a`), then there may be a problem with the modules loading correctly at boot.

The following is an example of using the `nmcli` tool to create an Intel® Omni-Path or InfiniBand* network device:

```
# nmcli con add type infiniband con-name ib0 ifname ib0 transport-mode connected
mtu 65520 ip4 192.168.100.10/24
```

NOTE

You need to configure both the Intel® Omni-Path HFI and IB devices using `nmcli`.

The `nmcli` command creates the ifcfg files in `/etc/sysconfig/network-scripts`.

Example of `/etc/sysconfig/network-scripts/ifcfg-ib0` created by `nmcli`:

```
CONNECTED_MODE=yes
TYPE=InfiniBand
BOOTPROTO=none
DEFROUTE=yes
IPV4_FAILURE_FATAL=no
IPV6INIT=no
IPV6_AUTOCONF=no
IPV6_DEFROUTE=no
IPV6_FAILURE_FATAL=no
NAME=ib0
UUID=eb1de50f-48f0-4f79-aafb-9eb3d8024bc0
DEVICE=ib0
ONBOOT=yes
MTU=65520
IPADDR=192.168.200.11
PREFIX=24
IPV6_PEERDNS=no
IPV6_PEERROUTES=no
```

Note that the syntax for configuring connected mode differs between RHEL and SLES. SLES configuration is covered in the next section.

After completing the device connection configurations for the HFI and IB devices the output of `ip a` should look similar to this:

```
4: ib0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 65520 qdisc pfifo_fast state UP
qlen 256
    link/infiniband 80:00:00:02:fe:80:00:00:00:00:00:00:00:11:75:01:01:57:42:38
brd 00:ff:ff:ff:ff:12:40:1b:80:01:00:00:00:00:00:00:ff:ff:ff:ff
    inet 192.168.200.11/24 brd 192.168.200.255 scope global ib0
       valid_lft forever preferred_lft forever
    inet6 fe80::211:7501:157:4238/64 scope link
       valid_lft forever preferred_lft forever
5: ib1: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 65520 qdisc pfifo_fast state UP
qlen 256
    link/infiniband 80:00:00:48:fe:80:00:00:00:00:00:00:00:02:c9:03:00:3d:5b:f1
brd 00:ff:ff:ff:ff:12:40:1b:ff:ff:00:00:00:00:00:00:ff:ff:ff:ff
    inet 192.168.100.11/24 brd 192.168.100.255 scope global ib1
       valid_lft forever preferred_lft forever
    inet 192.168.100.1/32 scope global ib1
```

```
       valid_lft forever preferred_lft forever
    inet6 fe80::202:c903:3d:5bf1/64 scope link
       valid_lft forever preferred_lft forever
```

Enable forwarding on the routers:

1. The coexisting routers need to able to forward packets. For a quick test of the functionality:

```
# echo 1 > /proc/sys/net/ipv4/conf/ib0/forwarding
# echo 1 > /proc/sys/net/ipv4/conf/ib1/forwarding
```

2. For persistence through a reboot, add the following entries to `/usr/lib/sysctl.d/00-system.conf`:

```
net.ipv4.conf.ib0.forwarding=1
net.ipv4.conf.ib1.forwarding=1
```

   This will limit IP forwarding to only the IB/Intel® Omni-Path interfaces.

2.5.1.4 Configuring Network Devices with yast2 for SLES*

The Intel® Omni-Path Basic installation will ask if you want to configure the IPoIB network scripts. You can accept this or use `yast2` to configure the interfaces or create the files manually in `/etc/sysconfig/network/`.

The following is an example of the `/etc/sysconfig/network/ifcfg-ib0` file created by `yast2`:

```
BOOTPROTO='static'
BROADCAST=''
ETHTOOL_OPTIONS=''
IPADDR='192.168.100.10/24'
IPOIB_MODE='connected'
MTU='65520'
NAME=''
NETWORK=''
REMOTE_IPADDR=''
STARTMODE='auto'
```

2.5.2 10/40/100 Gbit Ethernet

The Intel® Omni-Path Fabric Linux router can be used with 10/40/100 Gbit Ethernet devices. The following command uses the `nmcli` command to configure a 10/40/100 Gbit Ethernet device:

```
# nmcli con add type ethernet con-name eth2 ifname eth2 mtu 9000 ip4
192.168.100.20/24
```

NOTE

Intel recommends an MTU of 9000 (jumbo frame) for 10/40/100 Gbit Ethernet.

2.6 Client Configuration

To test the routers a minimum of one Intel® Omni-Path node and one IB node (as standalone representatives of their respective fabrics, apart from the subnet manager node) need to be configured and connected to the same switched environment that the routers are on.

The network configuration on RHEL* and SLES* uses the same setup as mentioned previously to configure the Intel® Omni-Path HFI or the IB HCA network devices.

NOTE

The router node should not be running the Intel® Omni-Path Fabric Suite Fabric Manager or the IB subnet manager.

2.6.1 Intel® Omni-Path Node Build

The Intel® Omni-Path node in this environment should be built from the Intel® Omni-Path IFS package. This provides all the packages from the Intel® Omni-Path Basic build and adds the Intel® Omni-Path Fabric Suite Fabric Manager. If the Fabric Manager is running on the Intel® Omni-Path switch then the installation of the Intel® Omni-Path Basic software will be sufficient.

2.6.2 IB Node Build

The IB node network interface works directly from a base build of RHEL* or SLES*, and a subnet manager can be added with the addition of the opensm package if the subnet manager is not running on the InfiniBand* switch.

For both the Intel® Omni-Path and IB nodes, a static route needs to be added to the network configuration for the client nodes to use the Linux router as a gateway to the opposing fabric. Based on the example network topology provided in this document, the routes will be added as described in the following sections.

2.6.2.1 Adding Static Routes

On the IB node:

```
# ip route add 192.168.200.0/24 via 192.168.100.1
```

This can be written to `/etc/sysconfig/network-scripts/route-<interface>` to keep the routing tables persistent through a reboot.

Example:

```
# cat /etc/sysconfig/network-scripts/route-ib0
192.168.100.0/24 via 192.168.200.1 dev ib0
```

NOTE

Sometimes the above route is NOT added at system boot. This is due to the Intel® Omni-Path and IB interfaces not being fully operational on the fabric when the NetworkManager service starts due to delays to an active state from the fabric or subnet managers, causing the static route configuration to fail. A workaround is to add the following to `/etc/rc.d/rc.local` on RHEL* or `/etc/rc.d/after.local` on SLES*:

1. Set execute permissions on /etc/rc.d/rc.local:

```
# chmod +x /etc/rc.d/rc.local
```

   The `rc.local` file:

```
sleep 30 #Delays adding the static route addition until the HFI/HCA is up.
/usr/sbin/ip route add 192.168.100.0/24 via 192.168.200.1
logger –p info “STATIC ROUTED ADDED”  # Use this to confirm that the rc.local
sript runs."
```

2. Start the rc-local service before rebooting the system to activate the service.

```
# systemctl start rc-local.service
```

At this point, with both routers up and connected to the fabrics you should be able to ping nodes from the Intel® Omni-Path fabric subnet through the router to the IB fabric subnet.

2.7 Router Redundancy/Failover with VRRP v3

Virtual Router Redundancy Protocol (VRRP) v3 is used to provide fault-tolerance, failover and load-balancing on the routers.

VRRP v3 is used in this documentation. VRRP v3 is part of the `keepalived` service. VRRP v3 RFC 5798 is available here: https://tools.ietf.org/html/rfc5798.

2.7.1 Prerequisites

Prior to using router redundancy/failover with VRRP, the failover routes *need to be specified on the routers themselves*. Then when one router fails, communication will transfer to the HFI on the other router.

Example

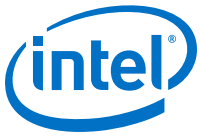
Two routers, each containing two HFIs:

router01:

- ib0, 192.168.1.1 (intended to fail over to 192.168.1.3)
- ib1, 192.168.1.2 (intended to fail over to 192.168.1.4)

router02:

- ib0, 192.168.1.3



- ib1, 192.168.1.4

The following routes are specified on the routers:

router01:

```
ip route add 192.168.1.3 dev ib0
ip route add 192.168.1.4 dev ib1
```

router02:

```
ip route add 192.168.1.1 dev ib0
ip route add 192.168.1.2 dev ib1
```

2.7.2 keepalived Installation

`keepalived` must be version 1.3.6 or later. If `keepalived` is less than 1.5.0, then see Using keepalived Version 1.3.5 on page 64 for additional configuration information regarding gratuitous arps.

Installing `keepalived` on RHEL using yum:

```
# yum install keepalived
# systemctl enable keepalived
```

Specific versions of `keepalived` can be installed by using rpm. The user will need to know the list of package dependencies for `keepalived`.

Additionally, the user may download and compile any version of `keepalived`. The sources are available at http://www.keepalived.org/download.htm.

To compile `keepalived` from source, it is possible that other libraries used by `keepalived` may need to be installed. Instructions and documentation are available at www.keepalived.org.

2.7.3 IP Routing Guidance

In scenarios where two IPoFabric interfaces on the same host (for example, ib0/ib1) are configured on the same subnet, care must be taken to define the Linux routing tables for the desired behavior. This is because Linux may always use ib0 to send return packets even if the communication request is initiated over ib1. This also should be configured for proper VRRP operation.

```
HostA:
ib0: 192.168.1.1
ib1: 192.168.2.1

HostB:
ib0: 192.168.1.2
ib1: 192.168.2.2
```

If a netmask, such as 255.255.0.0, is defined for the above hosts, communication over ib1 between the hosts may not work correctly. In this scenario, define the following routes on both hosts:

```
ip route add 192.168.1.0/24 via ib0
ip route add 192.168.2.0/24 via ib1
```

Alternatively, you can add these definitions to route-ib0/ib1 files.

Related Links

Adding Static Routes on page 27

2.7.4 Ensuring keepalived Successfully Launches at Boot Time

Due to the delay caused by the subnet manager/fabric manager services in bringing the IB and Intel® Omni-Path devices to an active state, VRRP may fail at boot time. The solution is to add this line to the `/etc/systemd/system/keepalived` xml file:

```
"After=NetworkManager-wait-online.service"
```

This delays the start of `keepalived` service until the network devices are in an active state.

Enable the `NetworkManager-wait-online.service`:

```
# systemctl enable NetworkManager-wait-online.service
```

The following is an example of the `keepalived.xml` file:

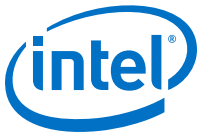
```
[Unit]
Description=LVS and VRRP High Availability Monitor
After=syslog.target network.target
After=NetworkManager-wait-online.service

[Service]
Type=forking
KillMode=process
EnvironmentFile=-/etc/sysconfig/keepalived
ExecStart=/usr/sbin/keepalived $KEEPALIVED_OPTIONS
ExecReload=/bin/kill -HUP $MAINPID
[Install]
WantedBy=multi-user.target
```

Place the `keepalived.xml` file in `/etc/systemd/system/`.

2.7.5 Configuring keepalived

There are two files involved in configuring `keepalived`.

The first is /etc/keepalived/keepalived.conf. This is the configuration file that defines keepalived keyword settings. See the man page `keepalived.conf(5)` for complete information.



The second file is `/etc/sysconfig/keepalived`. This file sets the runtime options for `keepalived`. Two useful options are `-D` for detailed log messages and `-P` to run only the VRRP service. These are both recommended settings when building and testing the routers for fault-tolerance.

The basic setup for fault-tolerance and automatic failover is to configure one router as the master for both subnet virtual IPs and the other as a backup. In the event the master router fails, the backup router will immediately assume the master role.

Another option is to configure the routers so that they split the traffic, with one router as the master on one subnet and the second router as the backup on the second subnet. This setup has the potential to reduce load on an individual router if traffic is going back and forth between the fabrics.

Below is a configuration example for a simple master/backup router pair.

```
/etc/keepalived/keepalived.conf for router1 (master):

vrrp_instance VI_1 {
       state MASTER
       interface ib0
       virtual_router_id 1
       priority 250
       authentication {
           auth_type PASS
           auth_pass password
       }
       virtual_ipaddress {
              192.168.200.1
       }
}
vrrp_instance VI_2 {
    interface ib1
    state MASTER
    virtual_router_id 2
    priority 250
    authentication {
        auth_type PASS
        auth_pass password
    }
    virtual_ipaddress {
        192.168.100.1
    }
}
```

The following is an example of `/etc/keepalived/keepalived.conf` for router2 (backup):

```
vrrp_instance VI_1 {
       state BACKUP
       interface ib0
       virtual_router_id 1
       priority 100
       authentication {
           auth_type PASS
           auth_pass password
       }
       virtual_ipaddress {
              192.168.200.1
       }
}


vrrp_instance VI_2 {
```

```
    interface ib3
    state BACKUP
    virtual_router_id 2
    priority 100
    authentication {
        auth_type PASS
        auth_pass password
    }
    virtual_ipaddress {
       192.168.100.1
     }
  }
}
```

NOTE

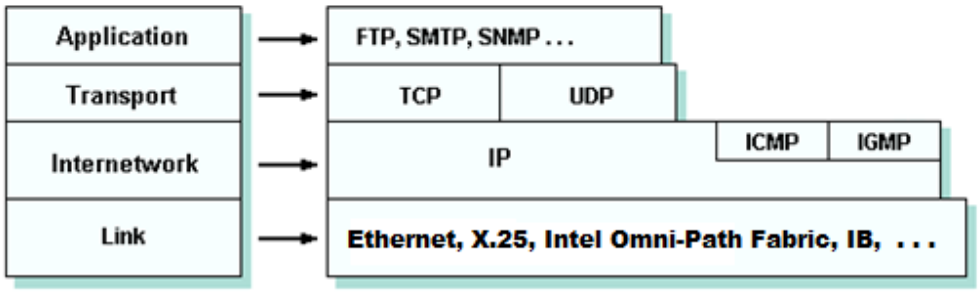
In the above `keepalived.conf` file the auth_type value is PASS, which transmits the password in plain text. Another option is auth_type AH, which uses IPSEC and transmits an encrypted password. Discussion of overall VRRP protocol security is outside of scope of this document. Please see RFC 5798 for more information at https://tools.ietf.org/html/rfc5798.

2.8 Performance Benchmarking and Tuning

2.8.1 Concerning Maximum Transmission Units (MTUs)

There are Maximum Transmission Units (MTUs) that apply both to the datagrams of the Internet or IP layer and to the packets of the Link layer on either side of the IP router. The following diagram shows the layered TCP/IP protocol stack.

Figure 7. Mapping of Network Layers and IP Stack Layers



The transport layer performs host-to-host communications on either the same or different hosts and on either the local network or remote networks separated by routers. The Transmission Control Protocol (TCP) provides flow-control, connection establishment, and reliable transmission of data.

The internetwork layer has the task of exchanging datagrams (a.k.a. packets) across network boundaries. This layer defines the addressing and routing structures used for the TCP/IP protocol suite. The primary protocol in this scope is the Internet Protocol (IP), which defines IP addresses. IPoIB and IPoFabric provide network interfaces at this layer.

The link layer, implemented by the Intel® Omni-Path fabric on one side of the IP router, and by InfiniBand* or Ethernet or another non-OPA technology on the other side of the IP router, includes the protocols used to describe the local network topology and the interfaces needed to transmit packets to next-neighbor hosts or switches.

2.8.1.1 Path MTU

In the context of IP routing, it is the effective size of the largest transmission unit that can be sent end to end without fragmentation between two communicating internet protocol (IP) endpoints that we wish to maximize for efficiency of bulk data transfers. By definition, the effective path MTU between communicating endpoints cannot be larger than the MTU of either endpoint, and this information is exchanged during the SYN message of TCP connections, so the use of larger MTU sizes on all communicating endpoints is encouraged.

The largest link level MTU size supported on the IPoFabric interface depends on if IPoFabric is operating in datagram mode or connected mode.

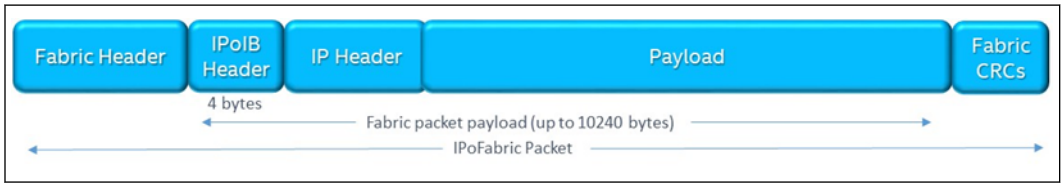
2.8.1.2 Datagram Mode MTU

Configuring "jumbo" Ethernet datagram sizes, and operating the router in datagram mode can help with router efficiency.

For IPoFabric bandwidth benchmarks, a prerequisite for good throughput performance is to utilize a link layer MTU that at least matches the link layer MTU of what is on the non-OPA side of the router. For example, when an IP router is routing from OPA to an Ethernet link with a configured jumbo MTU size of 9000 bytes, it is beneficial to increase the IPoFabric MTU size on the OPA link as well, greater than its default of 2044 bytes.

Prior to the OPA IFS 10.9 software release, the largest size configurable was 4092 bytes, constrained by what is defined for InfiniBand interfaces. With the IFS 10.9 release, the datagram mode MTU for IPoFabric can be specified larger, up to what fits in an OPA packet. Accommodating the 4 byte IPoIB header, IPoFabric datagram MTU sizes of up to 10236 bytes can now be configured. Details on how to configure this larger datagram mode IPoFabric MTU size are provided in the IPoFabric section of the *Intel® Omni-Path Fabric Performance Tuning User Guide*.

Figure 8. Larger MTU Size Enables Full OPA-size Payload in IPoFabric Datagram Packets



2.8.1.3 Connected Mode MTU

Another way to use a larger effective MTU size for transfers across the OPA fabric is to utilize IPoFabric in connected mode. When operating in connected mode, data is segmented and re-assembled, as necessary, to deliver a payload from one OPA node

to another across the fabric. The underlying segmentation and re-assembly allows for an IP layer MTU to be specified larger than what can fit in a single OPA packet, up to the maximum size supported by the networking stack, which is 65520 bytes.

When operating IPoFabric in Connected Mode, it is recommended to be configured with a 65520 byte IP layer MTU. This can be beneficial for bulk data transfers over an OPA fabric; however, note that connected mode becomes less efficient for bulk data transfers using smaller segment sizes.

With the introduction of Accelerated IPoFabric for datagram mode in IFS 10.9, it may be advantageous to use datagram mode when implementing an IP router (unless the routing is to IPoIB also in connected mode).

NOTE

The Intel® Omni-Path and IB link level fabric interfaces on an IP router are independently configurable interfaces. Connected mode for IPoFabric on the Intel® Omni-Path fabric HFI interface can be set independently from the mode of IPoIB configured on the the IB HCA interface, and independently of ib_ipoib's ipoib_enhanced setting that may have to be turned off to allow connected mode on the IPoIB interface.

The sections on Router Configuration on page 23 and Client Configuration on page 27 discuss methods for configuring IPoIB to automatically start upon reboot and how to configure the nodes for IPoIB Connected Mode operation.

One way to do a quick check if you are in connected mode with the corresponding IP MTU size is to cat two files:

```
# cat /sys/class/net/ib0/mode
connected
# cat /sys/class/net/ib0/mtu
65520
```

2.8.2 BIOS Tuning

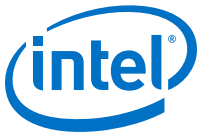
Refer to BIOS Settings on page 20 for information about the desired BIOS settings for good performance for client or router nodes.

Here we just emphasize that for the router nodes, Intel recommends setting Hyper-Threading Technology to Disabled.

2.8.3 OPA Interface Tuning

2.8.3.1 Datagram Mode or Connected Mode - Which is Better?

Prior to the IFS 10.9 software release, performance with connected mode generally offered better aggregate bandwidth performance than with datagram mode. With the introduction of Accelerated IPoFabric for datagram mode in the IFS 10.9 release, datagram mode may be better performing than connected mode for some applications. The answer to which mode is better for a fabric may differ based on the application. For IP routing, with the advent of Accelerated IPoFabric, datagram mode may be preferred.



By default, the OPA Fast Fabric tools will install and configure connected mode for new installations. Installers may want to try a few target application runs in datagram mode after the initial installation completes in connected mode to help decide what mode would be best to leave the system in for operations.

2.8.3.2 Accelerated Datagram Mode

Accelerated IPoFabric is enabled by default for all OPA IPoFabric interfaces on a host. An IPoFabric interface operates in datagram mode under a variety of conditions. Datagram mode communications are used for broadcast and multicast sends and receives, and when either, or both of a pair of communicating nodes does not support connected mode. In datagram mode, the MTU is determined by the lesser of the MTU configured on the interface, or the multicast MTU size defined by the Fabric Manager for the virtual fabric being used for IPoFabric communication on the fabric.

To increase the MTU allowed for datagram communications beyond the default 2044 value, the multicast MTU size for IPoFabric networking needs to be increased from the Fabric Manager, as described in the *Intel® Omni-Path Fabric Performance Tuning User Guide*.

If a user needs to revert to non-accelerated default IPoFabric for some reason, it is possible to do so by setting `options hfi1 ipoib_accel=0` in the local `/etc/modprobe.d/hfi1.conf` configuration file, and then reloading the hfi1 driver.

To make this change persistent across reboots it is advised to recreate initrd using:

```
# dracut -f
```

The current status of Accelerated IPoFabric module parameter for an IPoFabric interface can be determined by examining the `/sys/module/hfi1/parameters/ipoib_accel` value, where a return value of 1 indicates available and enabled.

2.8.3.3 Accelerated IPoFabric Tunings

An IP router is a specialized application. When configuring an IP router for best performance, there are some recommendations for IPoFabric that differ from what is described in the performance tuning guide. Those differences are described here.

It is not recommended to set XPS and RPS on the router if you are using Datagram Mode with Accelerated IPoFabric, which already employs core spreading.

IP router nodes will have interrupts to process from InfiniBand* or Ethernet interfaces in addition to interrupts to process from OPA interfaces. To conserve IRQs on the router, adjustments are recommended to be made to krcvqs and num_sdma.

When Accelerated IPoFabric is enabled (as discussed in the *Intel® Omni-Path Fabric Performance Tuning User Guide*) allocated kernel receive queues are not used the way they are in connected mode, and their count should be minimal to free cores for other purposes (set krcvqs=1).

In addition, for proper operation of the router, ensure that num_sdma is limited to 8 instead of the default 16, for the IP router's IPoFabric interface. There are a few options for how this limit can be configured. Options include:

- Enable the Networking virtual fabric at the FM. This is an advanced technique that results in IPoFabric traffic being segregated into its own virtual fabric with its own service level separate from MPI and other traffic types on the fabric. Enabling this second virtual fabric with its own QoS requirements results in a subset of the available sdma engines being reserved at each interface for IPoFabric communications.
- Locally configure the num_sdma module parameter and set it equal to 8. (This is the preferred approach, unless multiple virtual fabrics are enabled at the Fabric Manager.)

```
$ cat /etc/modprobe.d/hfi1.conf
options hfi1 krcvqs=1 num_sdma=8
```

2.8.3.4 Connected Mode Tunings

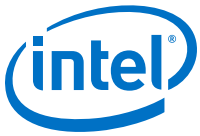
Starting with the IFS 10.9 release, Intel recommends considering operating IP router in datagram mode with Accelerated IPoFabric. If the IP router is to be operated in connected mode, then it is still recommended to design how the cores should be used on the IP router, and to enable RPS as a way to further spread some of the IPoFabric processing load across those available cores.

Receive Packet Steering (RPS) and Transmit Packet Steering (XPS) distribute packet processing among multiple CPUs to improve the performance of the network stack when the underlying network interface is not inherently multi-queue. Such as is the case with default IPoIB used in connected mode, and when accelerated IPoFabric has been disabled.

This section discusses how to apply RPS and XPS in an IP router for both OPA and non-OPA interfaces (e.g., Ethernet or InfiniBand*). For more information about RPS and XPS, please refer to the "Scaling in the Linux Networking Stack" document available at: https://www.kernel.org/doc/Documentation/networking/scaling.txt

The following rules are recommended to define a mask of CPUs used for XPS and RPS:

- RPS settings
  - RPS settings for OPA interfaces should be configured based on a recommendation in the *Intel® Omni-Path Fabric Performance Tuning User Guide*: RPS should be configured to direct the workload to the same CPUs that are servicing the receive context interrupts.
  - RPS settings for non-OPA cards (e.g., Ethernet or InfiniBand*) should be configured to forward packets to CPUs where an OPA card is connected, but not used by receive context interrupts or receive and send completion queue processes.
  - RPS for OPA and non-OPA interfaces should not be configured when using Accelerated IPoFabric and Datagram Mode.
- XPS settings
  - No performance gain is expected for settings of OPA XPS and non-OPA XPS. The recommendation is to keep them as default values, 0.



For more details of how to set the RPS, please refer the *Intel® Omni-Path Fabric Performance Tuning User Guide*. In case of non-OPA cards with multiple receive and transmit queues, e.g., Ethernet, the RPS and XPS settings must be applied to each queues in the cards. In general, RPS and XPS settings for a device <dev> with queue <n> can be found in the following paths:

```
RPS: /sys/class/net/<dev>/queues/rx-<n>/rps_cpus
XPS: /sys/class/net/<dev>/queues/tx-<n>/xps_cpus
```

Where <dev> is the name of the Ethernet or OPA interface in Linux.

Example: Accelerated IPoIB Disabled

NOTE

The following example is for illustrative purposes only. Your conditions may vary.

To determine CPUs whose IRQs OPA uses for kctxt, please use the method mentioned below. For more details, please refer to the Driver IRQ Affinity Assignments section of the *Intel® Omni-Path Fabric Performance Tuning User Guide*.

First, identify the CPU socket where the adapter is connected using the following command:

```
cat /sys/class/net/<dev>/device/local_cpus
```

For example, with 2 18-core CPUs where an OPA adapter is located in socket 1 and a non-OPA adapter is at socket 0, the output of the command is as follows:

```
opa:     00000000,00000000,00000000,00000000,0000000f,fffc0000
non-opa: 00000000,00000000,00000000,00000000,00000000,0003ffff
```

Next, identify CPUs for kernel receive contexts of the OPA adapter (hfi1_0 in this example) by typing the following command:

```
dmesg | grep hfi1_0 | grep IRQ | grep ctxt
[79.710022] hfi1 0000:81:00.0: hfi1_0: IRQ: 259, type RCVCTXT ctxt 0 -> cpu: 18
[79.718650] hfi1 0000:81:00.0: hfi1_0: IRQ: 260, type RCVCTXT ctxt 1 -> cpu: 19
[79.727298] hfi1 0000:81:00.0: hfi1_0: IRQ: 261, type RCVCTXT ctxt 2 -> cpu: 20
[79.735938] hfi1 0000:81:00.0: hfi1_0: IRQ: 262, type RCVCTXT ctxt 3 -> cpu: 21
[79.744592] hfi1 0000:81:00.0: hfi1_0: IRQ: 263, type RCVCTXT ctxt 4 -> cpu: 22
[79.753220] hfi1 0000:81:00.0: hfi1_0: IRQ: 264, type RCVCTXT ctxt 5 -> cpu: 23
```

For this example, system has been configured with 5 OPA hfi1 kernel receive contexts (parameter krcvqs=5 to hfi1 kernel module). RCVTXT ctxt0 is for management traffic, which is assigned to CPU 18. The other 5 receive contexts are for workloads and they are assigned to CPU 19-23.

In addition to CPU sockets and CPUs for kernel receive contexts, CPUs used for processing receive and send completion queues need to be identified to map the non-OPA RPS. Those completion queue processing CPUs can be identified by launching qperf or iperf in Measuring Baseline IP Performance on page 42 in order to send packets from compute nodes to a storage server, and by observing CPU behaviors of the IP router. While running the benchmark, a CPU with high utilization among non-krcvq CPUs means that the CPU is assigned for the receive completion queue

processing (`OPA rx-cq` in the table below), while the next core is typically used for the send completion queue processing (`OPA tx-cq` in the table below); one can also verify the assignment of `OPA tx-cq` by launching benchmarks to reverse direction, i.e., by sending packets from storage servers to compute nodes. Please note that, when the number of compute nodes increases, these CPUs can be highly utilized and may become the bottleneck of the router.

For this example, CPU 27 and 28 are used by receive and send completion queue processes, respectively.

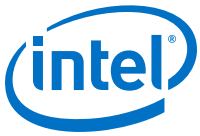
Once CPU sockets, CPUs for the receive contexts and CPUs for receive and send completion queue processes are identified, XPS and RPS can be configured based on the rules described above. For this example, if `<non-OPA_dev>` and `<OPA_dev>` are the name for non-OPA and IPoFabric interfaces respectively, OPA RPS is assigned to CPU 19-23, while non-OPA RPS is assigned to CPU 24-26, 29-35 with the following commands:

```
echo “0,00f80000” > /sys/class/net/<OPA_dev>/queues/rx-0/rps_cpus
echo “f,e7000000” > /sys/class/net/<non-OPA_dev>/queues/rx-<n>/rps_cpus
```

The table below summarizes all RPS assignments (marked as bold, last column), and the additional information (non-bold, 3rd column), such as receive context interrupts and TX/RX completion queue processes.

Table 1. CPUs Assignment in Example of IP Router (Accelerated IPoFabric Disabled)

Core	Socket	Load	
18	1	OPA kcrvq 0	
19	1	OPA kcrvq 1	**OPA RPS**
20	1	OPA kcrvq 2	**OPA RPS**
21	1	OPA kcrvq 3	**OPA RPS**
22	1	OPA kcrvq 4	**OPA RPS**
23	1	OPA kcrvq 5	**OPA RPS**
24	1		**Non-OPA RPS**
25	1		**Non-OPA RPS**
26	1		**Non-OPA RPS**
27	1	OPA rx-cq	
28	1	OPA tx-cq	
29	1		**Non-OPA RPS**
30	1		**Non-OPA RPS**
31	1		**Non-OPA RPS**
32	1		**Non-OPA RPS**
33	1		**Non-OPA RPS**
34	1		**Non-OPA RPS**
35	1		**Non-OPA RPS**



To make the above tuning persist after reboot, refer to Persisting Ethernet Tuning on page 41.

2.8.4 Ethernet Interface Tuning

IP router performance when routing to Ethernet may be improved with tunings applied to the Ethernet interface. The Ethernet tunings recommended in this section mainly focus on allowing packets to be forwarded with minimal processing as soon as they arrive at the Ethernet interface of an IP router.

NOTE

Please note that tuning recommended in this section may increase the load on CPUs by disabling offloading capabilities of the Ethernet interface or by increasing the number of system interrupts. These high-CPU utilizations may be acceptable for a router, where routing packets is the primary workload for the node. However, using these tunings for other nodes, such as storage servers and compute nodes, may significantly reduce performance of the system. Therefore, tunings described here should only be taken into consideration for routers.

2.8.4.1 TCP/IP Segmentation and Receive Offloads

Various Ethernet offloading capabilities implemented in the Linux* network stack are known to improve the performance of TCP/IP endpoints on a network. However, when they are used in an IP router with an OPA adapter, they can create unnecessary overhead to process IPoFabric in the OPA adapter.

Full description of the offloading techniques used for Linux kernel, can be found at https://www.kernel.org/doc/Documentation/networking/segmentation-offloads.txt. Disabling some of them may increase performance of an IP router. Consider disabling the following technologies:

- Generic Segmentation Offload - GSO
- Generic Receive Offload - GRO
- TCP Segmentation Offload - TSO
- Partial Generic Segmentation Offload - GSO_PARTIAL

To turn them off, please use the command below:

```
# ethtool -K <dev> gro off tso off gso off tx-gso-partial off
```

NOTE

The operation described above may negatively impact the performance of an Ethernet interface with small MTU (e.g., 1500). Please verify that the tuning works as intended on your target network.

To make the above tuning persist after reboot, refer to Persisting Ethernet Tuning on page 41.

2.8.4.2 Adaptive RX and TX

Similar to Ethernet offloading, enabling interrupt coalescence parameters on Ethernet may introduce extra delays to IP packets to be delivered to the OPA adapter. These parameters are sometimes enabled by default in Linux. To check the coalescence parameters of the Ethernet, use the command below:

```
# ethtool -c <inet>
Coalesce parameters for enp6s0f1:
Adaptive RX: on  TX: on
stats-block-usecs: 0
sample-interval: 0
pkt-rate-low: 0
pkt-rate-high: 0
[…]
```

Where <inet> is the name of the Ethernet interface in Linux.

Disabling the adaptive RX and TX and reducing their timeouts to 0 may increase the performance of the IP router:

```
sudo ethtool -C <inet> adaptive-tx off adaptive-rx off
sudo ethtool -C <inet> rx-usecs 0 tx-usecs 0
```

For further information about Adaptive RX and TX, please refer the Linux kernel documentation available at: https://www.kernel.org/doc/Documentation/networking/scaling.txt

To make the above tuning persist after reboot, refer to Persisting Ethernet Tuning on page 41.

2.8.4.3 IP Router with 40GbE Intel Interface (XL710)

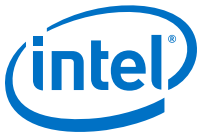
The Intel Ethernet flow director is described in detail in the Introduction to Intel Ethernet Flow Director and Memcached Performance whitepaper. The flow director has two main modes: the Externally Programmed mode and the automatic Application Targeted Routing (ATR) mode. Use the command below to check the current state:

```
# ethtool --show-priv-flags <inet>
Private flags for ens865f0:
MFP                     : off
LinkPolling             : off
flow-director-atr       : on
veb-stats               : off
hw-atr-eviction         : off
legacy-rx               : off
disable-source-pruning  : off
vf-true-promisc-support: off
```

It is recommended to disable ATR on the Intel Ethernet interfaces for better performance of the IP router. To disable ATR, use the following command:

```
# ethtool --set-priv-flags <inet> flow-director-atr off
```

Where <inet> is the name of the Ethernet interface in Linux.



To make the above tuning persist after reboot, refer to Persisting Ethernet Tuning on page 41.

2.8.5 Persisting Ethernet Tuning

Previous sections describe how to tune Ethernet settings and RPS/XPS settings for OPA and non-OPA interfaces. This following sections describe how to make configuration persistent so that the changes can still be applied after system reboot.

2.8.5.1 Persisting with Connected Mode

Perform the following steps using root privileges:

1. Ensure the Ethernet driver, OPA hfi1 driver and ipoib driver are configured to autostart.
2. Add the following lines to the file `/etc/rc.local`:

```
ethtool -K <ETH-port-name> gro off tso off gso off tx-gso-partial off
ethtool -C <ETH-port-name> adaptive-tx off adaptive-rx off
ethtool -C <ETH-port-name> rx-usecs 0 tx-usecs 0

# following line should be set for all rx queues for ETH interface
echo "00000xyz" > /sys/class/net/<ETH-port-name>/queues/rx-<n>/rps_cpus

# following line should be set for all rx queues for ETH interface
echo "00000abc" > /sys/class/net/<IPoIB-port-name>/queues/rx-0/rps_cpus


# following line should be set only for 40GbE Intel interface (XL710)
ethtool --set-priv-flags <ETH-port-name> flow-director-atr off
```

   Where:

   - <ETH-port-name> is name of Ethernet interface in Linux.
   - <IPoIB-port-name> is often ib0, but could be ib1, opa, opa_ib0, and so on.
   - "00000xyz", "00000abc" represents the hex mask (number) for your situation. Refer to Connected Mode Tunings on page 36 for instructions on how to construct the hex mask.
   - <n> is a RX queue id. The queue numbers starts with 0. Please note that an Ethernet interface may contain multiple RX queues, and the RPS mask settings should be applied to all RX queues in the interface.
3. Make sure `/etc/rc.local` script file is executable by issuing:

```
# chmod +x /etc/rc.local
```

4. Reboot to activate the changes.

2.8.5.2 Persisting with Datagram Mode and IPoFabric

Perform the following steps using root privileges:

1. Ensure the Ethernet driver, OPA hfi1 driver and ipoib driver are configured to autostart.

2. Add the following lines to the file `/etc/rc.local`:

```
ethtool -K <ETH-port-name> gro off tso off gso off tx-gso-partial off
ethtool -C <ETH-port-name> adaptive-tx off adaptive-rx off
ethtool -C <ETH-port-name> rx-usecs 0 tx-usecs 0

# following line should be set only for 40GbE Intel interface (XL710)
ethtool --set-priv-flags <ETH-port-name> flow-director-atr off
```

   Where:

   - <ETH-port-name> is name of Ethernet interface in Linux.
   - <IPoIB-port-name> is often ib0, but could be ib1, opa, opa_ib0, and so on.
   - <n> is a RX queue id. The queue numbers starts with 0. Please note that an Ethernet interface may contain multiple RX queues, and the RPS mask settings should be applied to all RX queues in the interface.

3. Make sure `/etc/rc.local` script file is executable by issuing:

```
# chmod +x /etc/rc.local
```

4. Reboot to activate the changes.

2.8.6 Measuring Baseline IP Performance

Before running storage benchmarks, which will run over IP protocols, it is important to measure the performance of IP over the Intel® Omni-Path and IB Fabrics: between Intel® Omni-Path clients, between storage servers, and between an Intel® Omni-Path client through the router to a storage server.

Point to point tools, such as qperf, or iperf, as described in the IPoFabric performance section of the *Intel® Omni-Path Fabric Performance Tuning User Guide* may be used for this purpose.

2.8.7 Router Saturation

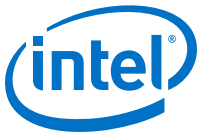
In order to identify the router saturation point, multiple `iperf3` can be launched simultaneously across multiple compute nodes and a single storage server. The total number of compute nodes to saturate a router depends on the configuration of the compute nodes. It is recommended to start testing with 4 compute nodes and 2 `iperf3` processes per compute node, and to increase the number of compute nodes up to 16 until the router is saturated.

Before launching `iperf3` client processes on the compute nodes, the storage server should have `iperf3` server processes to receive messages from the `iperf3` clients.

`iperf3` example for a server process (storage server):

```
iperf3 -s -1 -p <port_num> -A <cpu_affinity>
```

Once `iperf3` servers are launched at the storage server, `iperf3` clients can be launched across multiple compute nodes.



`iperf3` example for a client process (compute node):

```
iperf3 -c <server ipoib addr> -p <port_num> -t 20 -Z -l 1M -A <cpu_affinity>
```

In the above example, single `iperf3` process will run for 20 seconds with a 1M message size. While `iperf3` is running, the storage server should have a server process to receive messages from the client.

Note that each (client, server) `iperf3` pair should contain the same `<port_num>`, and the total number of `iperf3` server processes on the storage server should be equal to the total number of `iperf3` client processes across all compute nodes.

Once the saturation point is identified from multiple compute nodes to a storage server, it is recommended to test the saturation point of the reverse direction, i.e., from storage server to compute nodes, by launching `iperf3` servers on the compute nodes and `iperf3` clients on a storage server.

Router saturation points will vary depending on router usage. If the saturation points do not meet the requirements of the system, it is recommended to add more routers to improve performance.

3.0 LNet Router

3.1 Lustre.org

The primary documentation on setting up an LNet router can be found on Lustre.org. The documentation found on Lustre.org is maintained and updated by the Lustre* community as new developments are released and should be considered the primary configuration guide for LNet Routers and Lustre. This guide is meant to supplement the material found on Lustre.org and to provide some additional details on different aspects of configuration.

- http://wiki.lustre.org/LNet_Router_Config_Guide

This document includes an overview of an LNet router, additional details about configuring an LNet router, LNet router troubleshooting, LNet router server design considerations and a few configuration examples.

3.1.1 LNet Router Deployment Checklist

To efficiently make use of both Lustre.org and this guide, the following checklist aides in defining a set of logical steps to setup and deploy an LNet router. It assumes the Lustre file system is setup, and compute nodes and fabric are installed.

Design the router solution	See Designing LNet Routers to Connect Intel® OPA and InfiniBand* on page 50 or speak with your vendor of choice.
Setup of LNet Router Servers	Network card installation and OS installs
Install OPA/OFED Drivers and Lustre drivers	http://wiki.lustre.org/LNet_Router_Config_Guide%23Software_Installations
Setup and Configure LNet Router	http://wiki.lustre.org/LNet_Router_Config_Guide (and Overview of Configuring LNet on page 46) LNet configuration can be done in three ways: 1. Network Configuration by adding module parameters in `lustre.conf` 2. Dynamic network configuration using `lnetctl` command 3. Importing/exporting configurations using a YAML file format
(Optional) Perform fine grain routing in case of availability of multiple routes	http://wiki.lustre.org/LNet_Router_Config_Guide%23Fine-Grained_Routing
(Optional) Tune the LNet parameters	http://wiki.lustre.org/LNet_Router_Config_Guide#LNet_Tuning For Mellanox FDR/EDR cards based on the mlx5 driver in the router, make changes to the tunables.
Configure Lustre clients/mount Lustre	
Test LNet routing with LNet Self Test	http://wiki.lustre.org/LNET_Selftest
Troubleshooting	See LNet Troubleshooting on page 48.

The practical implementation examples at the bottom of the document also provide strong examples of LNet Router configuration.

3.2 Overview

Lustre file systems have the unique capability to run the same global namespace across several different network topologies. The LNet components of Lustre provide this abstraction layer. LNet is an independent project from Lustre and is used for other projects beyond the Lustre file system. LNet was originally based on the Sandia Portals project.

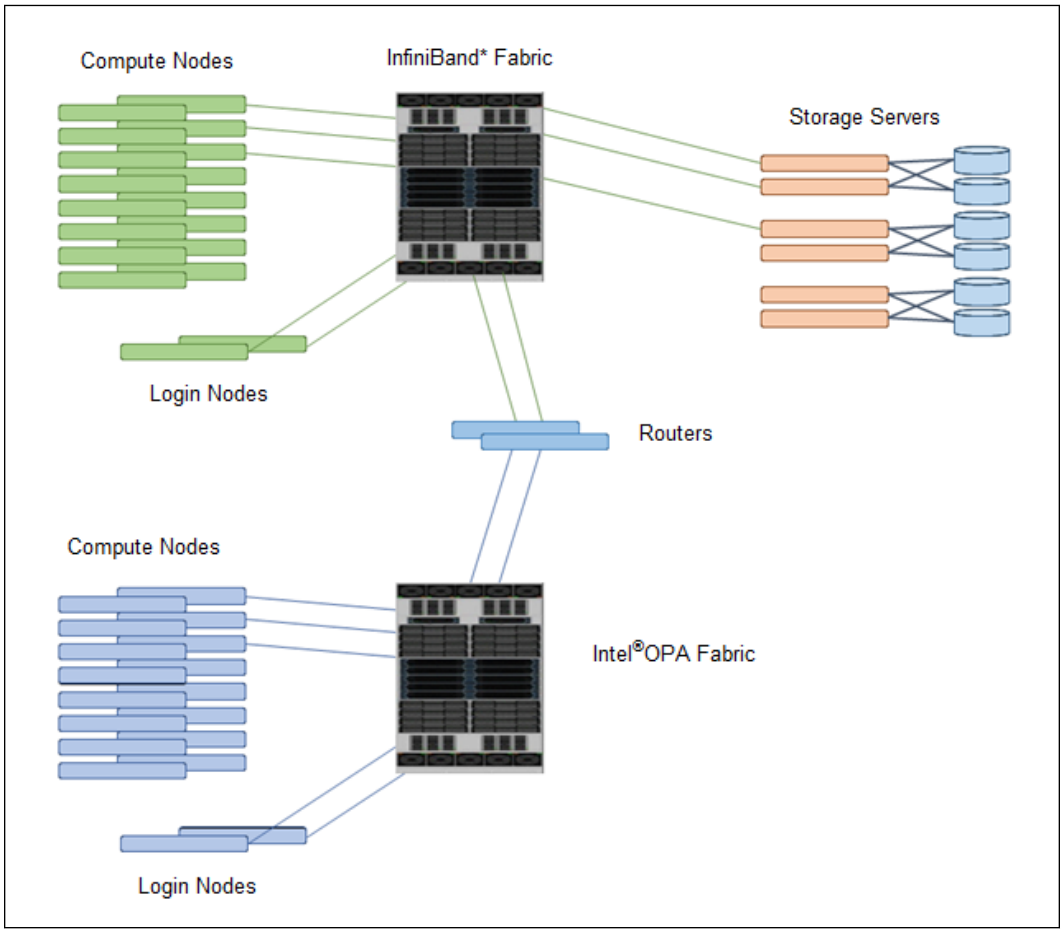
LNet can support Ethernet*, InfiniBand*, Intel® Omni-Path, legacy fabrics (ELAN and MyriNet*) and specific compute fabrics as Cray Gemini*, Aries*, and Cascade*.

LNet is part of the Linux kernel space and allows for full RDMA throughput and zero copy communications when available. Lustre can initiate a multi-OST read or write using a single Remote Procedure Call (RPC), which allows the client to access data using RDMA, regardless of the amount of data being transmitted.

LNet was developed to provide the maximum flexibility for connecting different network topologies using LNet routing. LNet's routing capabilities provide an efficient protocol to enable bridging between different networks, e.g., from Ethernet-to-InfiniBand, or the use of different fabric technologies such as Intel® Omni-Path Architecture (Intel® OPA) and InfiniBand*.

The following figure shows an example of how to connect an existing InfiniBand* network (storage and compute nodes) to new Intel® OPA compute nodes.

Figure 9. Heterogeneous Topology



3.3 Overview of Configuring LNet

To prepare for installation:

1. Obtain the PCIe bus number and device number of the Intel® Omni-Path/ InfiniBand* cards:

   For Intel® Omni-Path: `$ lspci | grep Omni-Path`

   For InfiniBand*: `$ lspci | grep Mellanox`

2. Check that the Intel® Omni-Path/InfiniBand* cards are installed and note the interface names. Use the PCIe bus number and device number to identify the correct card:

```
$ ls -la /sys/class/net/
```

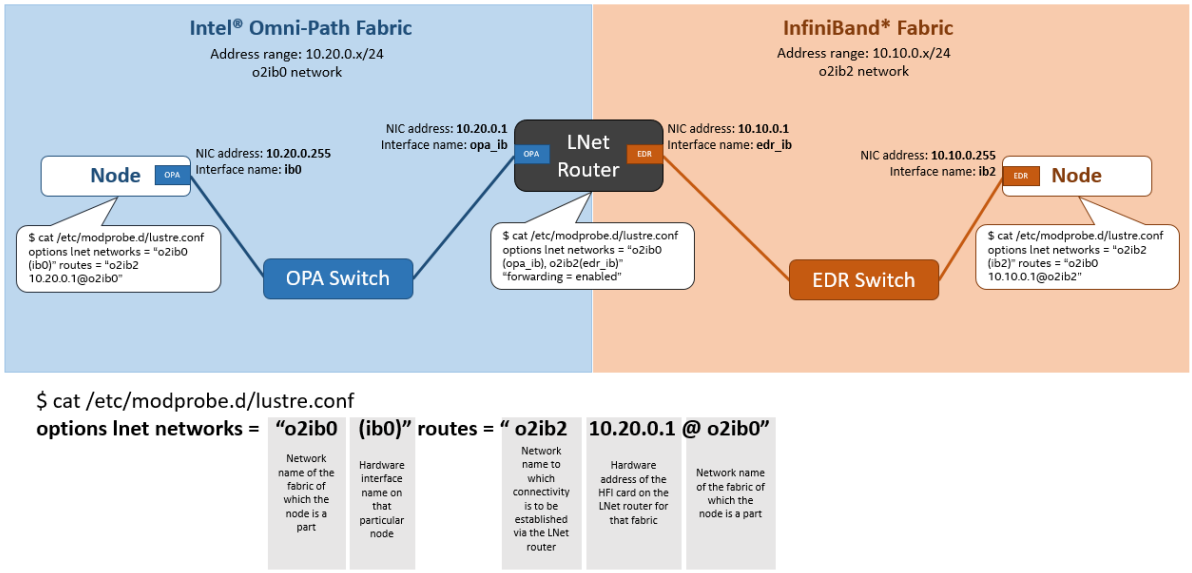
NOTE

Turbo mode on the LNet router must be disabled for consistent performance. CPU router frequency has almost negligible effect on the LNet router performance

An LNet router is a specialized Lustre client where only the LNet is running. An industry-standard, Intel-based server equipped with two sockets is appropriate for this role.

The Lustre file system is not mounted on the router, and a single LNet router can serve different file systems. In the context of LNet routing between two RDMA-enabled networks, in-memory zero copy capability is used to optimize latency and performance.

Figure 10. LNet Router



$ cat /etc/modprobe.d/lustre.conf

options lnet networks = "o2ib0 (Network name of the fabric of which the node is a part) **(ib0)" routes = " o2ib2** (Network name to which connectivity is to be established via the LNet router) **10.20.0.1 @ o2ib0"** (Hardware address of the HFI card on the LNet router for that fabric) (Network name of the fabric of which the node is a part) (Hardware interface name on that particular node)

Consider the simple example shown in the figure above, where:

- Storage servers are on a Mellanox-based InfiniBand fabric – 10.10.0.0/24
- Clients are on an Intel® OPA fabric - 10.20.0.0/24
- The router is between Intel® Omni-Path fabric and InfiniBand* fabric at 10.10.0.1 and 10.20.0.1
- For the purpose of setting up the lustre.conf files in this example, the Lustre network on the Intel® Omni-Path fabric is named `o2ib0` and the Lustre network on the InfiniBand* fabric is named `o2ib2`.

The network configuration on the **servers** (typically created in `/etc/modprobe.d/lustre.conf`) will be:

```
options lnet networks = "o2ib2 (ib2)" routes = "o2ib0 10.10.0.1@o2ib2"
```

The network configuration on the **LNet router** (typically created in `/etc/modprobe.d/lustre.conf`) will be:

```
options lnet networks = "o2ib0 (opa_ib), o2ib2(edr_ib)" "forwarding = enabled"
```

The network configuration on the **clients** (typically created in `/etc/modprobe.d/lustre.conf`) will be:

```
options lnet networks = "o2ib0 (ib0)" routes = "o2ib2 10.20.0.1@o2ib0"
```

Restarting LNet is necessary to apply the new configuration. Clients will mount the Lustre file system using the usual command line (assuming mgs1 and mgs2 are the IP addresses of the two Lustre servers hosting the MGS service on the o2ib0 network in the example):

```
mount -t lustre mgs1@o2ib0:/<file system name> /<mount point>
```

Lustre mount point is stored in `/etc/fstab`.

Reload Lustre module (for further debugging, refer to the LNet Dynamic Configuration section of the LNET Router Configuration Guide on the Lustre.org wiki to understand lnetctl commands):

```
$ modprobe -v lnet
```

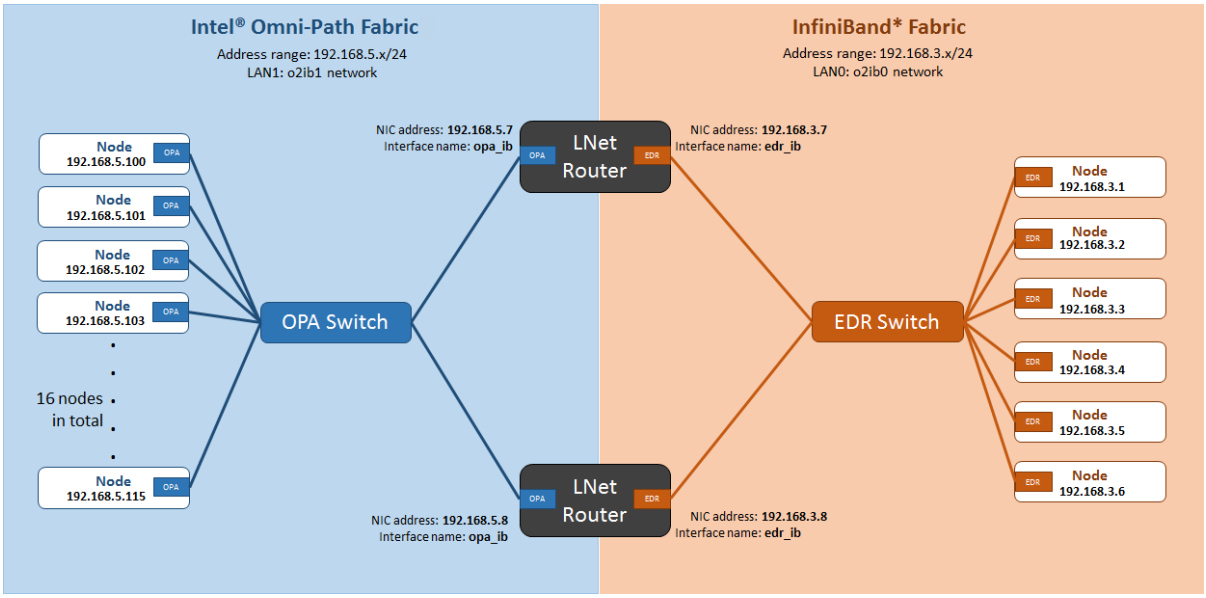
3.4 LNet Troubleshooting

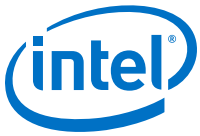
NOTE

The troubleshooting information in this section is valid for Lustre* version 2.11 and earlier.

LNet provides several metrics to troubleshoot a network.

Figure 11. Network Troubleshooting Using LNet





Referencing the figure above, consider the following configuration:

- Six Lustre servers are on LAN0 (o2ib0), a Mellanox-based InfiniBand network – 192.168.3.[1-6]
- Sixteen clients are LAN1 (o2ib1), an Intel® OPA network - 192.168.5.[100-115]
- Two routers on LAN0 and LAN1 at 192.168.3.[7-8] and 192.168.5.[7-8]

On each Lustre client we can see the status of the connections using the `/proc/sys/lnet/peers metric` file. This file shows all NIDs known to this node, and provides information on the queue state:

```
# cat /proc/sys/lnet/peers
nid                refs state  last   max   rtr   min    tx  min queue
192.168.5.8@o2ib1    4    up    -1     8     8     8     8  -505 0
192.168.5.7@o2ib1    4    up    -1     8     8     8     8  -473 0
```

Here, "state" is the status of the routers. In the case of a failure of one path, I/O will be routed through the surviving path. When both paths are available, RPCs will use both paths in round-robin.

Here, "max" is the maximum number of concurrent sends from this peer and "tx" is the number of peer credits currently available for this peer.

Notice the negative number in the "min" column. This negative value means that the number of slots on the LNet was not sufficient and the queue was overloaded. This is an indication to increase the number of peer credits and credits (see LNet Tuning).

Increasing the credits value has some drawbacks, including increased memory requirements and possible congestion in networks with a very large number of peers.

The status of the routing table can be obtained from the `/proc/fs/lnet/routes` file from a client:

```
# cat /proc/fs/lnet/routes
Routing disabled
net      hops priority   state router
o2ib        1        0      up 192.168.5.8@o2ib1
o2ib        1        0      up 192.168.5.7@o2ib1
```

The status of the routers can be verified from the `/proc/fs/lnet/routers` file from a client:

```
#cat /proc/fs/lnet/routers
ref rtr_ref alive_cnt state last_ping ping_sent deadline down_ni router
4      1        3    up     47         1         NA       0 192.168.5.7@o2ib1
4      1        1    up     47         1         NA       0 192.168.5.8@o2ib1
```

On each LNet router, the `/proc/sys/lnet/peers` metric shows all NIDs known to this node, and provides the following information (values are examples and not all information is shown):

```
#cat /proc/sys/lnet/peers
nid                refs  state   last   max   rtr   min    tx   min queue
192.168.3.4@o2ib0    1   up      165    8     8    -8     8   -15    0
192.168.3.1@o2ib0    1   up       47    8     8    -6     8    -8    0
192.168.3.6@o2ib0    1   up      165    8     8    -8     8   -15    0
192.168.3.3@o2ib0    1   down    115    8     8    -8     8   -12    0
```

```
192.168.3.5@o2ib0    1   up      153    8     8    -8     8    -8    0
192.168.3.2@o2ib0    1   up      83     8     8     8     8     7    0
192.168.5.113@o2ib1  1   up      65     8     8    -8     8    -6    0
192.168.3.104@o2ib1  1   down    9999   8     8    -8     8    -1    0
192.168.5.109@o2ib1  1   up      127    8     8    -4     8   -13    0
192.168.5.111@o2ib1  1   up      67     8     8    -8     8   -26    0
192.168.5.114@o2ib1  1   up      170    8     8    -3     8   -12    0
192.168.5.108@o2ib1  1   up      151    8     8    -4     8    -7    0
192.168.3.106@o2ib1  1   down    9999   8     8     4     8     4    0
192.168.5.101@o2ib1  1   up      58     8     8    -3     8    -9    0
192.168.5.103@o2ib1  1   up      178    8     8    -8     8   -14    0
192.168.5.102@o2ib1  1   up      63     8     8    -4     8   -18    0
.  .  .
```

In the output above, we can see some Lustre clients on LNet0 are down.

Credits are initialized to allow a certain number of operations. In the example in the above table, this value is 8 (eight), shown under the max column. LNet keeps track of the minimum number of credits ever seen over time showing the peak congestion that has occurred during the time monitored. Fewer available credits indicates a more congested resource.

The number of credits currently in flight (number of transmit credits) is shown in the "tx" column. The maximum number of send credits available is shown in the "max" column and that never changes. The number of router buffers available for consumption by a peer is shown in the "rtr" column.

Therefore, rtr - tx is the number of transmits in flight. Typically, rtr == max, although a configuration can be set such that max >= rtr. The ratio of routing buffer credits to send credits (rtr/tx) that is less than max indicates operations are in progress. If the ratio rtr/tx is greater than max, operations are blocking.

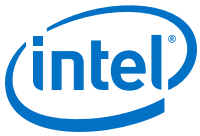
LNet also limits concurrent sends and number of router buffers allocated to a single peer, so that no peer can occupy all these resources.

Realtime statistics of the LNet router can be obtained using the `routerstat` command. Routerstat watches LNet router statistics. If no interval is specified, stats are sampled and printed only once; otherwise, stats are sampled and printed every interval. Output includes the following fields:

- M - msgs_alloc(msgs_max)
- E - errors
- S - send_count/send_length
- R - recv_count/recv_length
- F - route_count/route_length
- D - drop_count/drop_length

3.5 Designing LNet Routers to Connect Intel® OPA and InfiniBand*

The LNet router can be deployed using an industry standard server with enough network cards and the LNet software stack. Designing a complete solution for a production environment is not an easy task, but Intel is providing tools (LNet Self-Test) to test and validate the configuration and performance in advance.



The goal is to design LNet routers with enough bandwidth to satisfy the throughput requirements of the back-end storage. The number of compute nodes connected to an LNet router normally does not change the design of the solution.

The bandwidth available to an LNet router is limited by the slowest network technology connected to the router. Typically, Intel has observed a 10-15% decline in bandwidth from the nominal hardware bandwidth of the slowest card, due to the LNet router.

Multiple LNet Routers can be used to connect storage between different fabrics. This includes providing a degree of load balancing, failover and performance scaling. The performance scaling tends to be linear with large block IO and a balanced fabric, so adding additional routers increases throughput to the existing Lustre storage in a linear fashion.

In every case, Intel encourages validating the implemented solution using tools provided by the network interface maker and/or the LNet Self-Test utility, which is available with Lustre.

LNet routers can be congested if the number of credits (`peer_credits` and `credits`) are not set properly. For communication to routers, not only a credit and peer credit must be tuned, but a global router buffer and peer router buffer credit are needed.

To design an LNet router in this context, we need to consider the following topics:

- Hardware design and tuning
- Software compatibility

3.5.1 Hardware Design and Tuning

When designing an LNet router between two different network technologies such as Mellanox InfiniBand and Intel® OPA, one should consider that LNet was developed taking advantage of the RDMA zero copy capability. This makes the LNet router extremely efficient.

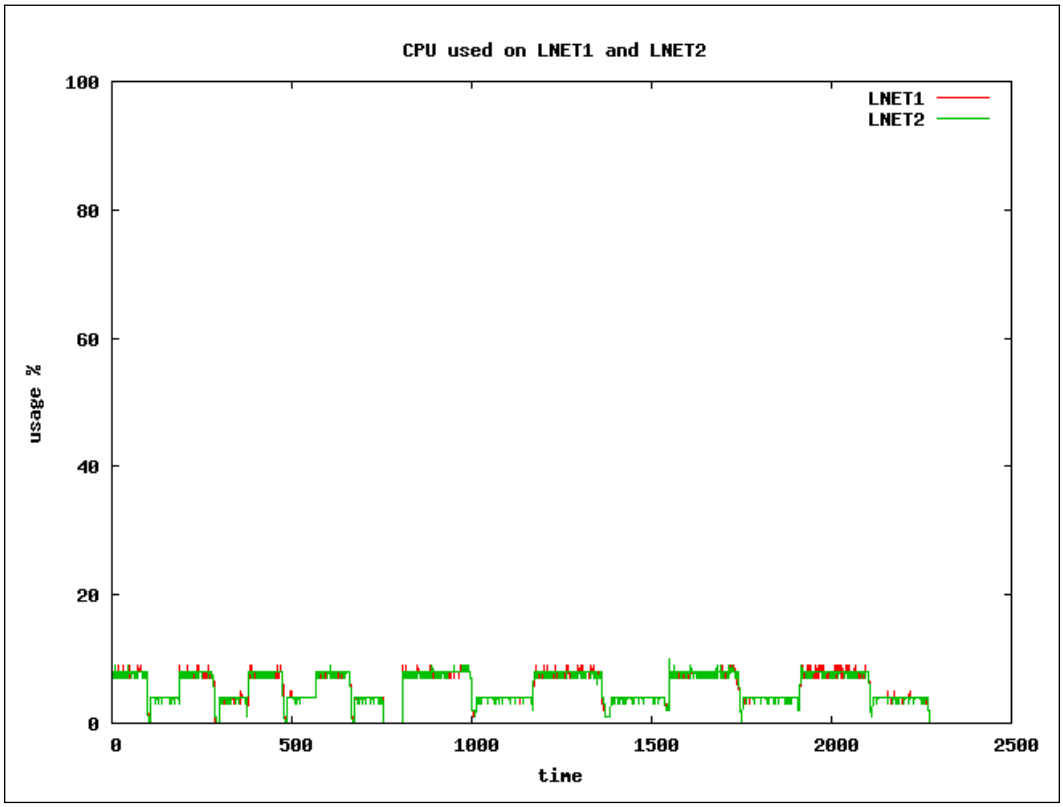
To achieve higher performance from Intel® OPA in a Lustre file system, one must tune the LNet stack as described in LNet Tuning and in the *Intel® Omni-Path Fabric Performance Tuning User Guide*.

3.5.2 CPU Selection

Generally speaking, the CPU performance is not critical for the LNet router code, and the recent SMP affinity implementation enables the LNet code to scale on NUMA servers.

The following figure shows the CPU utilization of two LNet routers configured for load balancing and routing an Intel® OPA client network and a Mellanox FDR storage network during a large IOR test. The activity between the two routers is completely specular and balanced. The CPU utilization is below 10% to sustain a FDR card. The CPU activity is two times during WRITE compared to READ. Both routers were equipped with two Intel® Xeon® Processors (E5-2697 v2) clocked at 2.7 GHz. Further testing changing the frequency speed of the CPU was run and proved the CPU frequency had no effect on the LNet router performance.

Figure 12. CPU Utilization for Two LNet Routers Routing Between an Intel® OPA Network and a Mellanox FDR Network



To obtain higher performance, Intel suggests turning off the Hyper-Threading Technology and Frequency Scaling capabilities of the CPU (see table below).

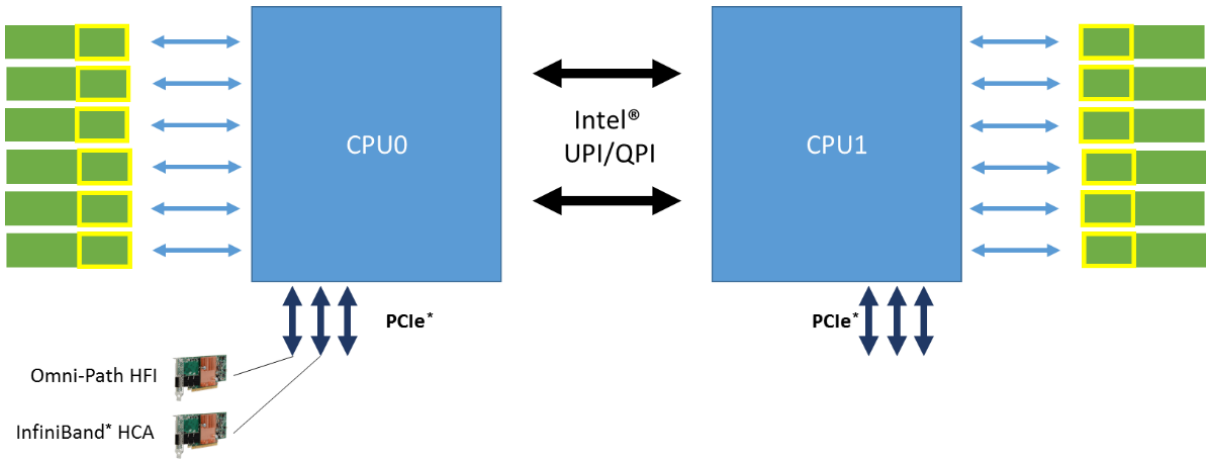
Tests using different CPU frequencies also indicate LNet Routing is not clock frequency sensitive. Selecting an Intel® Xeon® Processor with a moderate number of cores and moderate frequency is recommended.

Table 2. LNet Router CPU Tuning

Hardware	Recommendation
CPU	Cost effective current processor such as the Intel® Xeon® Gold 6130 Processor
Hyperthreading	OFF
CPU Frequency Scaling	DISABLED

It is important to select the right PCIe* slot in the server for the Intel® OPA and IB cards to avoid long distance paths in the NUMA architecture. For an optimal configuration ensure the network cards are in PCIe busses connected to the same CPU socket. See the following figure.

Figure 13. PCIe* Slot Allocation



3.5.3 Memory Considerations

An LNet router uses additional credit accounting when it needs to forward a packet for another peer:

- Peer Router Credit: This credit manages the number of concurrent receives from a single peer and prevent single peer from using all router buffer resources. By default this value should be 0. If this value is 0 LNet router uses `peer_credits`.
- Router Buffer Credit: This credit allows messages to be queued and select non data payload RPC versus data RPC to avoid congestion. In fact, an LNet Router has a limited number of buffers:
  - `tiny_router_buffers` – size of buffer for messages of <1 page size
  - `small_router_buffers` – size of buffer for messages of 1 page in size
  - `large_router_buffers` – size of buffer for messages >1 page in size

These LNet kernel module parameters can be monitored using the `/proc/sys/lnet/buffers` file and are available per CPT:

```
pages count credits     min
    0   512     512     503
    0   512     512     504
    0   512     512     497
    0   512     512     504
    1  4096    4096    4055
    1  4096    4096    4050
    1  4096    4096    4048
    1  4096    4096    4072
  256   256     256     244
  256   256     256     248
  256   256     256     240
  256   256     256     246
```

Negative numbers in the "min" column above indicate that the buffers have been oversubscribed; we can increase the number of router buffers for a particular size to avoid stalling.

The memory utilization of the LNet router stack is caused by the Peer Router Credit and Router Buffer Credit parameters. An LNet router with a RAM size of 32GB or more has enough memory to sustain very large configurations for these parameters. In every case, the memory consumption of the LNet stack can be measured using the `/proc/fs/sys/lnet/lnet_memused` metrics file.

Table 3. Sample Table

Hardware	Recommendation
RAM	32 GB
Technology	DDR3 or DDR4 ECC

3.5.4 Software Compatibility

This section discusses compatibility considerations for the software stack to be used:

- The Intel® Fabric Suite (IFS) for Intel® OPA supports RHEL* 7.4.
- The Mellanox OFED 3.x stack is supported from Lustre* software version 2.4 or later.

3.6 Practical Implementations

This section covers two of the most common network interface configurations:

- Example 1: Legacy storage with InfiniBand* card ConnextX-3/IB/4 connected to new compute nodes using Intel® OPA.
- Example 2: New storage with Intel® OPA connected to legacy compute nodes on InfiniBand* card ConnextX-3/IB/4.

NOTE

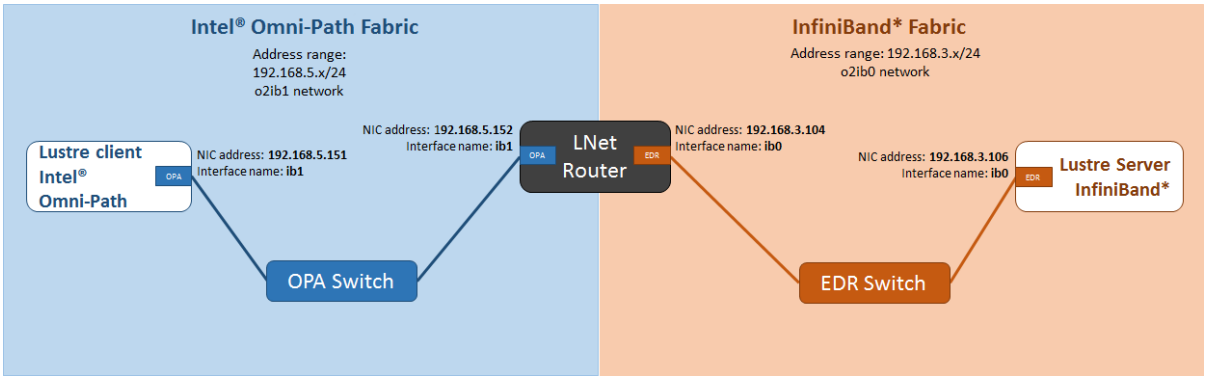
Throughout these examples, IP addresses are examples only. In these two example configurations, we made the assumption that all the components can be upgraded. Please consult an Intel Lustre specialist for non-destructive methods to upgrade Lustre. We will use as much as possible the Dynamic LNet Configuration (DLC) technology.

3.6.1 Example 1: Legacy Storage with InfiniBand* Card Connected to New Compute Nodes Using Intel® OPA

The following figure shows this simplified network topology:

- a Lustre client equipped with an Intel® Omni-Path fabric card
- an LNet router equipped with an Intel® OPA card and an InfiniBand* card
- a legacy Lustre server equipped with an InfiniBand* card

Figure 14. Network Topology for Example 1 Configuration



To achieve this configuration, perform the following procedures:

1. Upgrade all Lustre servers, Lustre clients and LNet routers to Lustre* software version 2.10 or later.
2. Perform the steps in the Configure Lustre* Clients (Example 1) on page 55.
3. Perform the steps in the Configure LNet Routers (Example 1) on page 56.
4. Perform the steps in the Configure Lustre* Servers (Example 1) on page 57.

3.6.1.1 Configure Lustre* Clients (Example 1)

The following commands are based on the topology in Figure 14 on page 55.

```
#modprobe lnet
#lnetctl lnet configure
#lnetctl net add --net o2ib1 --if ib1
#lnetctl route add --net o2ib0 --gateway 192.168.5.152@o2ib1
#lnetctl net show --verbose
net:
    - net: lo
      nid: 0@lo
      status: up
      tunables:
          peer_timeout: 0
          peer_credits: 0
          peer_buffer_credits: 0
          credits: 0
          CPT: "[0,0,0,0]"
    - net: o2ib1
      nid: 192.168.5.151@o2ib1
      status: up
      interfaces:
          0: ib1
          lnd tunables:
              peercredits_hiw: 64
              map_on_demand: 32
              concurrent_sends: 256
              fmr_pool_size: 2048
              fmr_flush_trigger: 512
              fmr_cache: 1
      tunables:
          peer_timeout: 180
          peer_credits: 128
          peer_buffer_credits: 0
          credits: 1024
          CPT: "[0,0,0,0]"
```

```
#lnetctl route  show --verbose
route:
    - net: o2ib
      gateway: 192.168.5.152@o2ib1
      hop: 1
      priority: 0
      state: up
```

To make the configuration permanent:

```
#lnetctl export > /etc/sysconfig/lnet.conf
#echo "o2ib0: { gateway: 192.168.5.152@o2ib1 }" > /etc/sysconfig/lnet_routes.conf
#systemctl enable lnet
```

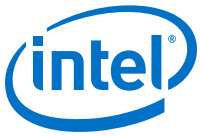
3.6.1.2 Configure LNet Routers (Example 1)

By default, Lustre* software will deploy the following `ko2iblnd` configuration (`/etc/modprobe.d/koblnd.conf`) to optimize any existing Intel® OPA cards:

```
alias ko2iblnd-opa ko2iblnd
options ko2iblnd-opa peer_credits=128 peer_credits_hiw=64
credits=1024 concurrent_sends=256 ntx=2048 map_on_demand=32
fmr_pool_size=2048 fmr_flush_trigger=512 fmr_cache=1

install ko2iblnd /usr/sbin/ko2iblnd-probe
```

Some of the above parameters are not compatible with InfiniBand* cards, so we will use DLC to set per-card parameters using the following procedure:

```
#modprobe lnet

#lnetctl lnet configure
#lnetctl net add --net o2ib1 --if ib1
#lnetctl net add --net o2ib0 --if ib0
#lnetctl set routing 1
#lnetctl net show --verbose
net:
    - net: lo
      nid: 0@lo
      status: up
      tunables:
          peer_timeout: 0
          peer_credits: 0
          peer_buffer_credits: 0
          credits: 0
          CPT: "[0,0,0,0]"
    - net: o2ib1
      nid: 192.168.5.152@o2ib1
      status: up
      interfaces:
          0: ib1
          lnd tunables:
              peercredits_hiw: 64
              map_on_demand: 32
              concurrent_sends: 256
              fmr_pool_size: 2048
              fmr_flush_trigger: 512
              fmr_cache: 1
      tunables:
          peer_timeout: 180
          peer_credits: 128
          peer_buffer_credits: 0
          credits: 1024
          CPT: "[0,0,0,0]"
```



```
  - net: o2ib
    nid: 192.168.3.104@o2ib
    status: up
    interfaces:
        0: ib0
        lnd tunables:
            peercredits_hiw: 64
            map_on_demand: 32
            concurrent_sends: 256
            fmr_pool_size: 2048
            fmr_flush_trigger: 512
            fmr_cache: 1
    tunables:
        peer_timeout: 180
        peer_credits: 128
        peer_buffer_credits: 0
        credits: 1024
        CPT: "[0,0,0,0]"
```

To edit the configuration and to make it permanent:

```
#lnetctl export > /etc/sysconfig/lnet.conf
```

InfiniBand* cards based on the mlx5 driver are not compatible with the `map_on_demand=32` and other parameters.

For the InfiniBand* card, edit the `lnet.conf` file and add the following new parameters (bolded below).

```
- net: o2ib
   nid: 192.168.3.104@o2ib
   status: up
   interfaces:
       0: ib0
       lnd tunables:
          peercredits_hiw: 64
          map_on_demand: 32
          concurrent_sends: 256
          fmr_pool_size: 2048
          fmr_flush_trigger: 512
          fmr_cache: 1
   tunables:
       peer_timeout: 180
       peer_credits: 128
       peer_buffer_credits: 0
       credits: 256
       CPT: "[0,0,0,0]"
```

To enable the configuration at startup:

```
#systemctl enable lnet
```

3.6.1.3 Configure Lustre* Servers (Example 1)

The Lustre servers should already be configured, however we need to change the configuration in order to add the routing path to the Intel® OPA network and enable the new Intel® OPA clients through the LNet routers.

Remove any LNet configuration normally in `/etc/modprobe.d/lustre.conf`

```
#modprobe lnet

#lnetctl lnet configure
#lnetctl net add --net o2ib0 --if ib0
#lnetctl route add --net o2ib1 --gateway 192.168.3.104@o2ib0
#lnetctl net show --verbose
net:
    - net: lo
      nid: 0@lo
      status: up
      tunables:
          peer_timeout: 0
          peer_credits: 0
          peer_buffer_credits: 0
          credits: 0
          CPT: "[0,0,0,0]"
    - net: o2ib
      nid: 192.168.3.106@o2ib
      status: up
      interfaces:
          0: ib0
          lnd tunables:
              peercredits_hiw: 4
              map_on_demand: 0
              concurrent_sends: 8
              fmr_pool_size: 512
              fmr_flush_trigger: 384
              fmr_cache: 1
      tunables:
          peer_timeout: 180
          peer_credits: 8
          peer_buffer_credits: 0
          credits: 256
          CPT: "[0,0,0,0]"

#lnetctl route  show --verbose
route:
    - net: o2ib1
      gateway: 192.168.3.104@o2ib
      hop: 1
      priority: 0
      state: up
```

To make the configuration permanent:

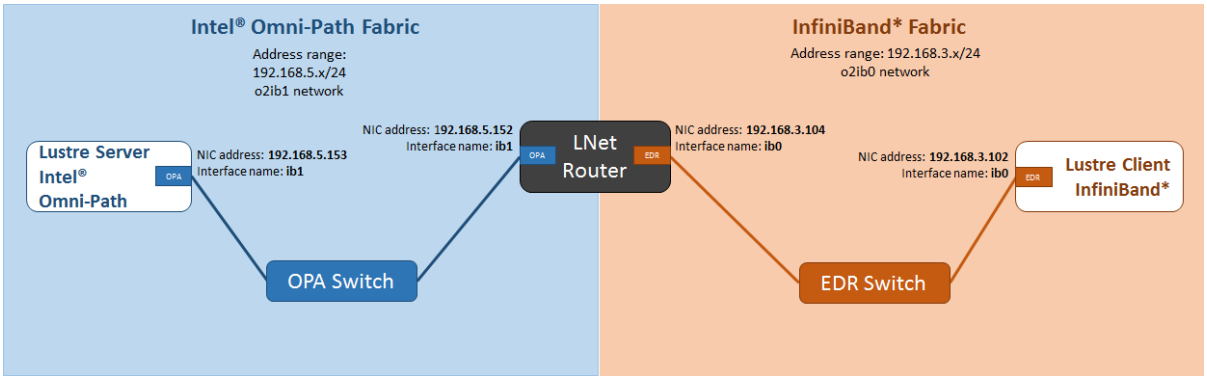
```
#lnetctl export > /etc/sysconfig/lnet.conf
#echo "o2ib1: { gateway: 192.168.3.104@o2ib0 }" > /etc/sysconfig/lnet_routes.conf
#systemctl enable lnet
```

3.6.2 Example 2: New Storage with Intel® OPA Connected to Legacy Compute Nodes on InfiniBand* Cards

The following figure shows another common example of a simplified network topology:

- a legacy Lustre* client connected via an InfiniBand* card
- an LNet Router implementing Intel® OPA and InfiniBand* cards
- a Lustre server connected via an Intel® OPA card.

Figure 15. Network Topology for Adding New OPA-Connected Servers



To achieve this configuration, perform the following procedures:

1. Upgrade all Lustre clients, Lustre servers and LNet Routers to Lustre* software 2.10 or later.

3.6.2.1 Configure Lustre* Clients (Example 2)

The following commands are based on the topology in Figure 15 on page 59. Reconfigure the clients after upgrade, removing the `/etc/modprobe.d/lustre.conf`

```
#modprobe lnet
#lnetctl lnet configure
#lnetctl net add --net o2ib0 --if ib0
#lnetctl route add --net o2ib1 --gateway 192.168.3.104@o2ib0
#lnetctl net show --verbose
net:
    - net: lo
      nid: 0@lo
      status: up
      tunables:
          peer_timeout: 0
          peer_credits: 0
          peer_buffer_credits: 0
          credits: 0
          CPT: "[0,0,0,0]"
    - net: o2ib
      nid: 192.168.3.102@o2ib
      status: up
      interfaces:
          0: ib0
          lnd tunables:
              peercredits_hiw: 4
              map_on_demand: 0
              concurrent_sends: 8
              fmr_pool_size: 512
              fmr_flush_trigger: 384
              fmr_cache: 1
      tunables:
          peer_timeout: 180
          peer_credits: 8
          peer_buffer_credits: 0
          credits: 256
          CPT: "[0,0,0,0]"

#lnetctl route  show --verbose
route:
    - net: o2ib1
```

```
      gateway: 192.168.3.104@o2ib
      hop: 1
      priority: 0
      state: up
```

To make the configuration permanent:

```
#lnetctl export > /etc/sysconfig/lnet.conf
#echo "o2ib1: { gateway: 192.168.3.104@o2ib0 }" > /etc/sysconfig/lnet_routes.conf

#systemctl enable lnet
```

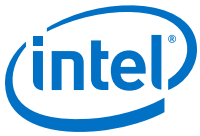
3.6.2.2 Configure LNet Routers (Example 2)

By default, Lustre* software will deploy the following `ko2iblnd` configuration (`/etc/modprobe.d/koblnd.conf`) to optimize any existing OPA card:

```
alias ko2iblnd-opa ko2iblnd
options ko2iblnd-opa peer_credits=128 peer_credits_hiw=64
credits=1024 concurrent_sends=256 ntx=2048 map_on_demand=32
fmr_pool_size=2048 fmr_flush_trigger=512 fmr_cache=1

install ko2iblnd /usr/sbin/ko2iblnd-probe
```

Some of the tunables are not compatible with InfiniBand* cards, so we will use DLC to set per-card tunables using the following procedure:

```
#modprobe lnet

#lnetctl lnet configure

#lnetctl net add --net o2ib1 --if ib1
#lnetctl net add --net o2ib0 --if ib0
#lnetctl set routing 1

#lnetctl net show --verbose
net:
    - net: lo
      nid: 0@lo
      status: up
      tunables:
          peer_timeout: 0
          peer_credits: 0
          peer_buffer_credits: 0
          credits: 0
          CPT: "[0,0,0,0]"
    - net: o2ib1
      nid: 192.168.5.152@o2ib1
      status: up
      interfaces:
          0: ib1
          lnd tunables:
              peercredits_hiw: 64
              map_on_demand: 32
              concurrent_sends: 256
              fmr_pool_size: 2048
              fmr_flush_trigger: 512
              fmr_cache: 1
      tunables:
          peer_timeout: 180
          peer_credits: 128
          peer_buffer_credits: 0
          credits: 1024
          CPT: "[0,0,0,0]"
```



```
    - net: o2ib
      nid: 192.168.3.104@o2ib
      status: up
      interfaces:
          0: ib0
          lnd tunables:
              peercredits_hiw: 64
              map_on_demand: 32
              concurrent_sends: 256
              fmr_pool_size: 2048
              fmr_flush_trigger: 512
              fmr_cache: 1
      tunables:
          peer_timeout: 180
          peer_credits: 128
          peer_buffer_credits: 0
          credits: 1024
          CPT: "[0,0,0,0]"
```

To edit the configuration and to make the configuration permanent, we will export it:

```
#lnetctl export > /etc/sysconfig/lnet.conf
```

InfiniBand* cards based on the mlx5 driver are not compatible with the `map_on_demand=32` and other parameters.

For the InfiniBand* card, edit the `lnet.conf` file and add the following new parameters (bolded below):

```
   - net: o2ib
     nid: 192.168.3.104@o2ib
     status: up
     interfaces:
         0: ib0
         lnd tunables:
            peercredits_hiw: 64
            map_on_demand: 32
            concurrent_sends: 256
            fmr_pool_size: 2048
            fmr_flush_trigger: 512
            fmr_cache: 1
     tunables:
         peer_timeout: 180
         peer_credits: 128
         peer_buffer_credits: 0
         credits: 256
         CPT: "[0,0,0,0]"
```

To enable the configuration at startup:

```
#systemctl enable lnet
```

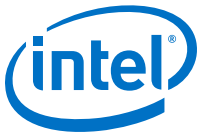
3.6.2.3 Configure Lustre* Servers (Example 2)

```
#modprobe lnet
#lnetctl lnet configure
#lnetctl net add --net o2ib1 --if ib1
#lnetctl route add --net o2ib0 --gateway 192.168.5.152@o2ib1
#lnetctl net show --verbose
net:
    - net: lo
      nid: 0@lo
```

```
      status: up
      tunables:
          peer_timeout: 0
          peer_credits: 0
          peer_buffer_credits: 0
          credits: 0
          CPT: "[0,0,0,0]"
    - net: o2ib1
      nid: 192.168.5.153@o2ib1
      status: up
      interfaces:
          0: ib1
          lnd tunables:
              peercredits_hiw: 64
              map_on_demand: 32
              concurrent_sends: 256
              fmr_pool_size: 2048
              fmr_flush_trigger: 512
              fmr_cache: 1
      tunables:
          peer_timeout: 180
          peer_credits: 128
          peer_buffer_credits: 0
          credits: 1024
          CPT: "[0,0,0,0]"

#lnetctl route show --verbose
route:
    - net: o2ib
      gateway: 192.168.5.152@o2ib1
      hop: 1
      priority: 0
      state: up
```

To make the configuration permanent:

```
#lnetctl export > /etc/sysconfig/lnet.conf
#echo “o2ib0: { gateway: 192.168.5.152@o2ib1 }” > /etc/sysconfig/lnet_routes.conf
#systemctl enable lnet
```



Appendix A Firewall Configuration for VRRP on RHEL

VRRP uses multicast address 224.0.0.18 for VRRP Advertisement messages between master and backup routers. This traffic can be observed with tcpdump.

The following is a tcpdump example of a VRRP advertisement.

Router1

```
# tcpdump -i ib0 host 224.0.0.18 -v -v
11:26:47.454956 IP (tos 0xc0, ttl 255, id 8057, offset 0, flags [none], proto
VRRP (112), length 40) 192.168.100.10 > vrrp.mcast.net: vrrp 192.168.100.10 >
vrrp.mcast.net: VRRPv2, Advertisement, vrid 2, prio 250, authtype simple, intvl
1s, length 20, addrs: 192.168.100.1 auth "password"
```

1. To allow VRRP Advertisements through the firewall on a Red Hat* Enterprise Linux* system using firewalld:

```
# firewall-cmd --permanent --zone=public --add-rich-rule='rule
family=ipv4 destination address=224.0.0.18 protocol value=ip
accept'
```

2. Reload firewalld to commit the change:

```
# firewall-cmd --reload
```

3. Verify the new firewall entry:

```
# firewall-cmd --list-all
public (default, active)
  interfaces: eth0 eth1 ib0 ib1
  sources: 224.0.0.8
  services: dhcpv6-client ssh
  ports:
  masquerade: no
  forward-ports:
  icmp-blocks:
  rich rules:
    rule family="ipv4" destination address="224.0.0.18" protocol value="ip"
accept
```

In a network where multiple router pairs are in use on the same subnets it is critical that the VRRP advertisements are isolated within each router pair. As can be seen in the above output from tcpdump a password is used in each advertisement. To ensure that advertisements from other router pairs do not interfere, a unique password should be used for each router pair.

Appendix B Using keepalived Version 1.3.5

When a `keepalived` node becomes master, it adds the master IP address to the specified interface and issues gratuitous ARP packets to tell the nodes connected to that interface to update their ARP Cache with the mac address of the new master.

keepalived prior to version 1.5.0 is not able to do this on an IPoIB interface, so a workaround is required. Version 1.3.5 of `keepalived` was tested with this workaround.

In the `/etc/keepalived/keepalived.conf` file, for each master instance block with an "`ibX`" named interface, add a notify_master line to issue the gratuitous ARPs. An example is shown below for illustrative purposes. Adjust as applicable for your installation.

```
vrrp_instance VI_1 {
         state MASTER
         interface ib0
         virtual_router_id 1
         priority 250
         authentication {
          auth_type PASS
          auth_pass password
         }
         virtual_ipaddress {
                    192.168.200.1
         }
         notify_master "/sbin/arping -q -U -c 5 -I ib0 192.168.200.1" root root
}
```